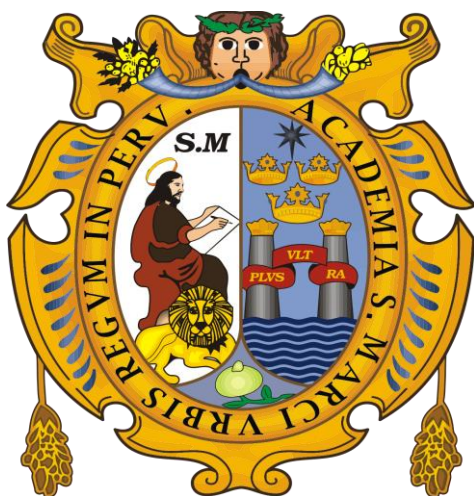


**UNIVERSIDAD NACIONAL DE SAN MARCOS**  
(Universidad del Perú, DECANA DE AMÉRICA)  
**Facultad de Ingeniería de Sistemas e Informática**  
E.A.P. de Ingeniería de Sistemas



**PROYECTO FINAL – SISTEMA DE RA**

**GRUPO 3**

**INTEGRANTES**

Alvarado Silva, Raúl Fernando	18200120
Paco Huamán, Juan Carlos	18200037

**CURSO**

**INTERNET DE LAS COSAS**

**DOCENTE**

**M. Sc. YESSICA ROSAS CUEVAS**

**Lima, Perú**

**2022-2**

## ÍNDICE

<b>Introducción</b>	<b>3</b>
Revisión del Estado de Arte	3
Planteamiento del Problema	5
Objetivos	5
Objetivo General	5
Objetivos Específicos	5
<b>Marco Teórico</b>	<b>6</b>
Sistema de Riego	6
Protocolo MQTT	6
Node-RED	7
Dashboard	7
<b>Componentes del Sistema de Riego</b>	<b>8</b>
<b>Implementación del Sistema</b>	<b>10</b>
Código Utilizado para el ESP8266	10
Explicación del Código	13
Datos sobre la conexión a internet	13
Función void setup()	13
Función void loop()	14
Conexión a Wifi	14
Conexión MQTT	15
Sistema de Riego Autónomo Implementado	15
Esquema de NODE-RED	16
<b>Resultados del Proyecto</b>	<b>17</b>
<b>Conclusiones</b>	<b>20</b>

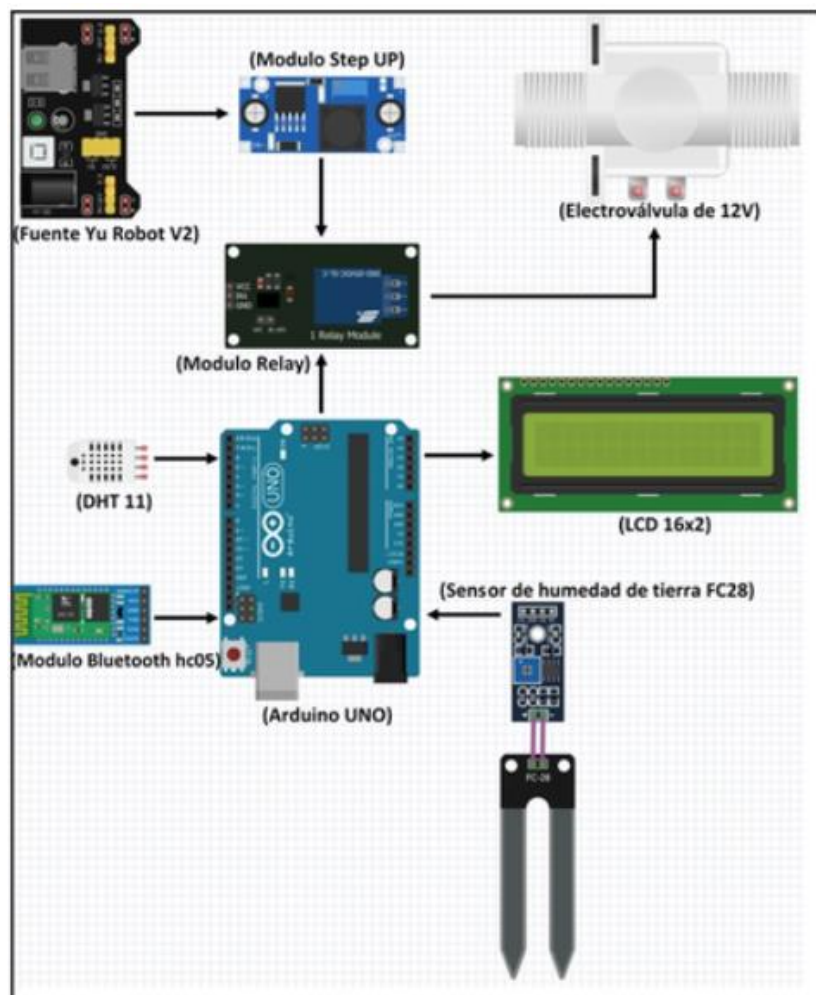
## Introducción

### Revisión del Estado de Arte

Se encontraron varias referencias para la elaboración del proyecto de sistema de riego autónomo. Donde mencionaremos a los siguientes trabajos:

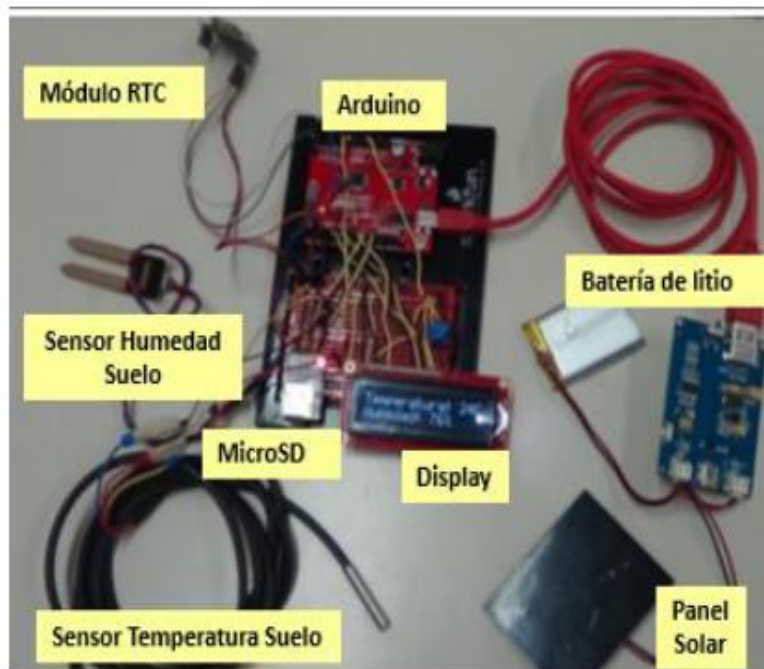
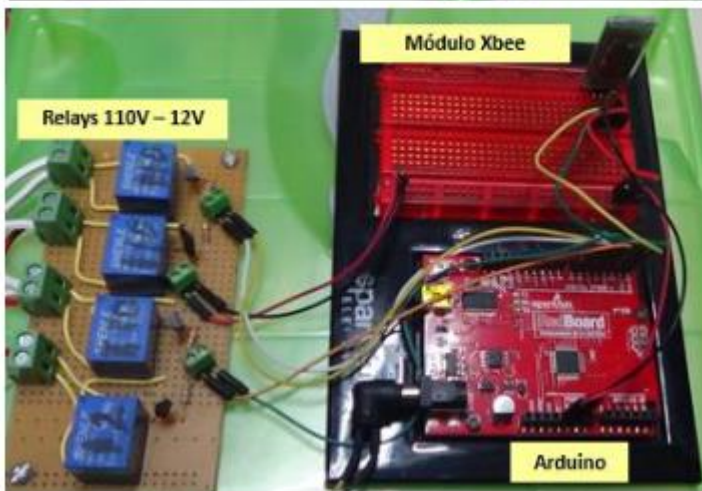
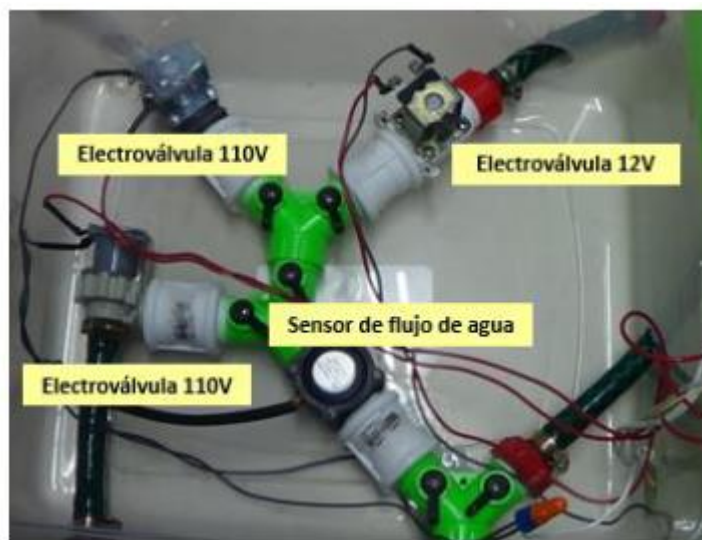
- **Sistema de Riego Automatizado con Arduino:**

Este trabajo publicado en la revista Espacios, en donde establece que la manutención de huertos domésticos resulta compleja, debido a que los jardines se secan por falta de hidratación. Para evitar esto, se plantea diseñar un sistema de riego automático, que combine soluciones de hardware y software libres, para medir la humedad de la tierra y el aire porque forman parte del ecosistema del huerto. A esta solución se le añadió un microcontrolador, que actúe como centro de operaciones para asegurar el suministro y la dosificación de agua para mantener hidratada una planta.



- **Sistema de Riego Autónomo basado en la Internet de las Cosas (IoT):**

El autor nos muestra el uso de tecnologías de bajo costo y de hardware – software libre como lo son: Raspberry Pi, Arduino, etc. Con los cuales se construyeron algunos modelos de predicción empleando los datos de estaciones meteorológicas. Por ello se implementó un sistema de Machine Learning para predicciones del clima. Además, se implementó una red de sensores inalámbricos para obtener información de las variables ambientales, donde estas variables son enviadas en formato JSON al nodo recolector Xbee. Y para mayor control del sistema se desarrolló una aplicación web, en la cual se pueda realizar el control de los actuadores como las electroválvulas y motobombas, como también obtener los valores de los nodos sensores.



## **Planteamiento del Problema**

En la actualidad, el mundo está afrontando varios problemas de alta gravedad; uno de ellos es el cambio climático. Este proyecto es considerado como una propuesta para enfrentar este problema y con ello obtener un mejor futuro. Para ello se establecieron los Objetivos de Desarrollo Sostenible (ODS), en el cual se tomará en cuenta el ODS N° 12: "Garantizar modalidades de consumo y producción sostenibles".

Debido a los sistemas domóticos, la automatización de varios aspectos en las viviendas se encuentra al alcance de todos, con ello surge la idea de un sistema de control de riego autónomo empleando tanto herramientas y tecnologías de IOT, donde se aplicará lo aprendido en clases y laboratorios de este curso.

## **Objetivos**

### ***Objetivo General***

Desarrollar e implementar un sistema de riego autónomo e inteligente utilizando y aplicando los conceptos y conocimientos de las clases y los laboratorios del curso de Internet de las Cosas.

### ***Objetivos Específicos***

- Diseñar el circuito con el sistema de control, los sensores y actuadores, además de construir los dashboard con la placa ESP8266 y la herramienta gratuita Node-RED.
- Aplicar el protocolo MQTT en el desarrollo del sistema de riego.
- Describir el desarrollo, funcionamiento y resultados de la aplicación de IoT para el sistema de riego inteligente propuesto.

## **Marco Teórico**

### **Sistema de Riego**

Según Anaya et al. (2016) un sistema de riego es visto como una herramienta para poder aumentar la capacidad productiva del sistema de producción, compuestos por distintos elementos que pueden ser mangueras o tuberías, válvulas de paso y motobombas, cuyos elementos permiten el paso del flujo de agua a través del sistema hacia un terreno o cultivo que se desea regar.

### **Protocolo MQTT**

Este es un protocolo de mensajería asíncrono caracterizado por ser liviano y flexible, es usado en los tipos de comunicación M2M para el desarrollo de aplicaciones de internet de las cosas. Su funcionamiento consiste en el intercambio de mensajes mediante el modelo de Publisher-Suscriber.

- **Modelo Publisher-Suscriber:** Funciona a través de la conexión a un nodo central, denominado como broker, tanto para la recepción y envío de mensajes se debe suscribir a un topic que son canales de comunicación en donde se va a disponer los mensajes. (Mahedero, 2020).
- **Cliente MQTT:** Un cliente MQTT puede ser tanto publicador como suscriptor, o ambos a la vez.
- **Broker MQTT:** Es un servidor que se encarga de la administración de los Topic, es decir, de la distribución de los mensajes a los receptores, el broker recibe los mensajes y verifica los clientes que están suscritos al topic y encamina los mensajes hacia ellos, además el broker brinda una medida de seguridad para autenticar la identidad de los clientes. (Mahedero, 2020).

- Topic: Es el canal de comunicación conectado al broker, también conocido como el tema del mensaje, es decir, de quien se quiere recibir el mensaje o a quien va dirigido ese mensaje y, además, sirve como un identificador al broker para poder enviar los mensajes a los clientes que se hayan suscrito a un topic. (Mahedero, 2020).

## **Node-RED**

Node-RED es una herramienta gratuita de desarrollo de código abierto para la integración de distintos dispositivos IoT, interfaces para la programación de aplicaciones y servicios en línea que ofrece IBM, está basada en JavaScript y fue creada en la plataforma Node.js, cuya plataforma proporciona un editor de flujo visual que está basado en el navegador, según M. Lekić y G. Gardašević (2018).

Node-Red permite que los desarrolladores puedan conectar nodos de entrada, salida y de procesamiento, de tal manera que se pueda crear flujos para el procesamiento de datos, enviar alertas o realizar un control sea como una suscripción a un topic MQTT, recibir un dato de un sensor y almacenarlos en una base de datos.

## **Dashboard**

De acuerdo con Viera et al. (2021), los dashboard son herramientas que te permiten agrupar, compartir y centralizar la información relevante de manera gráfica, de tal manera que se pueda realizar un seguimiento ya sea del estado de una empresa o un determinado proceso específico.

## Componentes del Sistema de Riego

**Tabla 1**

*Componentes del Sistema de Riego*

NOMBRE	FOTO	DESCRIPCIÓN
ESP8266		Microcontrolador con Wifi integrado el cual nos da acceso a una red. Permite conectar proyectos a una red Wifi y también elaborar los con Internet de las cosas (IOT).
DHT22		Permite monitorear temperatura y humedad relativa de forma precisa y sencilla. La salida suministrada es de tipo digital, no refiriéndose utilizar entradas analógicas.
SENSOR DE HUMEDAD DE SUELO		Permite obtener los valores de humedad de suelo de manera sencilla por medio de 2 electrodos resistivos.
BATERÍA 9V		Permite alimentar o proveer energía a proyectos con microcontroladores, especialmente en internet de las cosas (IoT).
RELAY 12V		Permite controlar el encendido/apagado de equipos de alta potencia (electrodomésticos).



<b>PORTA PILA Y PILA AA</b>		Para la alimentación del microcontrolador.
<b>BOMBA DE AGUA</b>		Mini Bomba de agua, nos sirve para el manejo del flujo de agua en la manguera.
<b>PROTOBOARD</b>		Herramienta para prototipar circuitos
<b>RESISTENCIAS 220Ω</b>		Permite alterar o modificar el paso de la corriente.
<b>LED RGB</b>		Componente electrónico que emite luz de diferentes colores.
<b>LED'S</b>		Componente electrónico que emite luz de un único color.
<b>CABLES</b>		Para realizar las conexiones entre los componentes.

## Implementación del Sistema

### Código Utilizado para el ESP8266

```
//Librerias Utilizadas
#include <ESP8266WiFi.h>//Libreria ESP8266 de Conexion a red
#include <PubSubClient.h>//Libreria Publish/Subscribe con un servidor MQTT
#include <DHTesp.h>//Libreria del sensor DHT
DHTesp dht; //Variable dht
// PIN de los led rgb
int Rojo = 5;
int Verde = 4;
int Azul = 0;
// PIN de la minibomba de agua
float bomba = 2;
// PIN del sensor de humedad de suelo
#define SensorPin A0
float humedad_suelo = 0;//Variable Humedad_Suelo

//Datos de la conexion a internet
const char* ssid = "MOVISTAR_5442"; //Nombre del router
const char* password = "4PRpDuXNfnuqrcAywsey"; // clave del router
const char* mqtt_server = "192.168.1.15"; // direccion Ip del Servidor MQTT

// Variables de conexion MQTT
WiFiClient espClient;
PubSubClient client(espClient);

// Variables globales
unsigned long lastMsg = 0;
#define MSG_BUFFER_SIZE (50)
char msg[MSG_BUFFER_SIZE]; // Se almacenar los datos de los sensores para
enviarselo al Node-Red
char dth22_temp[MSG_BUFFER_SIZE]; // DTH22 Temperatura
char dth22_hum[MSG_BUFFER_SIZE]; // DTH22 Humedad
char hum[MSG_BUFFER_SIZE]; // Humedad de suelo

void setup() {
    // Inicializando los PINES
    //ENTRADA
    pinMode(SensorPin, INPUT);
    //SALIDA
    pinMode(Rojo, OUTPUT);
    pinMode(Verde, OUTPUT);
    pinMode(Azul, OUTPUT);
    pinMode(bomba, OUTPUT);
    //Pin del sensor DHT22
    dht.setup(16, DHTesp::DHT22);
    //Iniciamos monitor de Serial
    Serial.begin(115200);
    //Iniciamos la Funcion WIFI
    inicializando_wifi();
    //Iniciamos el cliente
    client.setServer(mqtt_server, 1883);
    client.setCallback(callback);
}

void loop() {
    // Comprobamos la conexion
```

```

    if (!client.connected()) {
        reconnect();
    }
    client.loop();
    // Se envia los datos cada 25000 milisegundos
    long tiempo = millis();
    if(tiempo-lastMsg > 25000)
    {
        lastMsg = tiempo;
    }
    // Lectura de sensores
    TempAndHumidity valor = dht.getTempAndHumidity(); // Valores del sensor
DHT22
    humedad_suelo = analogRead(SensorPin); // Valor del sensor humedad de
suelo
    ledRGB(valor.temperature); // funcion para manipular el led rgb
    //Se imprimen los datos en el monitor serial
    Serial.println(" Humedad del ambiente: " + String(valor.humidity, 1)+
"%");
    Serial.println(" Temperatura: " + String(valor.temperature, 2)+ "C°\n");
    Serial.println("Humedad del suelo: " + String(humedad_suelo));
    delay(5000);
    // Conversion de datos para publicarlos
    sprintf(dth22_temp, "%3.2f", valor.temperature);
    sprintf(dth22_hum, "%3.2f", valor.humidity);
    sprintf(hum, "%3.2f", humedad_suelo);
    // Publicando datos de los sensores en NODE-RED
    client.publish("esp8266/temperatura", dth22_temp);
    client.publish("esp8266/humedad", dth22_hum);
    client.publish("esp8266/humedadSuelo", hum);
    //Funcion de encendido de bomba de agua
    enciendeBomba(humedad_suelo);
}

// Conexion a WIFI
void inicializando_wifi() {
    delay(10);
    // Constancia de conexion
    Serial.println();
    Serial.print("Conectado a ");
    Serial.println(ssid);
    // Iniciamos la conexion a la red
    WiFi.mode(WIFI_STA);
    WiFi.begin(ssid, password);
    // Constancia de Espera
    while (WiFi.status() != WL_CONNECTED) {
        delay(500);
        Serial.print(".");
    }
    //Inicializamos el generador de números pseudo aleatorios
    //Función micros(): Devolverá el número de microsegundos desde que la
placa empezó a ejecutar el programa
    randomSeed(micros());
    //Constancia de conexion realizada
    Serial.println("");
    Serial.println("WiFi conectado");
    Serial.println("IP address: ");
    Serial.println(WiFi.localIP());
}

//Conexion MQTT

```

```

void reconnect() {
    while (!client.connected()) {
        // Mensaje de constancia de conexion
        Serial.print("Intentando conexión Mqtt...");
        // Se crea un cliente ID
        String clientId = "ESP8266";
        // Iniciamos la conexion al MQTT
        if (client.connect(clientId.c_str() ) ) {
            // Mensaje de Conexion
            Serial.println("Conectado!");
        }
        else {
            // Mensajes de Error de Conexion
            Serial.print("falló :( con error -> ");
            Serial.print(client.state());
            Serial.println(" Intentamos de nuevo en 5 segundos");
            delay(5000);
        }
    }
}

// Lectura de datos que llegan despues de suscribir
void callback(char* topic, byte* payload, unsigned int length) {
    //Mensaje de constancia de recepcion de envio
    Serial.print("Mensaje llegado[");
    Serial.print(topic);
    Serial.print("] ");
    //Se muestra los mensajes enviados
    for (int i = 0; i < length; i++) {
        Serial.print((char)payload[i]);
    }
    Serial.println();
}

//Funcion de Encendido de Bomba de Agua
void enciendeBomba(double sensorHumedadSuelo)
{
    // Condicion de Encendido: El valor supere 700
    if(sensorHumedadSuelo>700){
        // Se enciende la bomba de agua por 15 segundos
        digitalWrite(bomba,LOW);
        delay(15000);
        digitalWrite(bomba,HIGH);
    }
}

//Funcion de LED RGB (Este cambiara segun la lectura de la Temperatura)
void ledRGB(double lectura)
{
    //Condiciones
    if(lectura < 18){ // Temperatura menor que 18°C: Color azul(Frio)
        digitalWrite(Rojo,0);
        digitalWrite(Verde,0);
        digitalWrite(Azul,255);
    }
    if(lectura >=18 || lectura < 25) // Temperatura entre que 18°C y 25°C:
    Color Verde(Ambiente)
    {
        digitalWrite(Rojo,0);
        digitalWrite(Verde,255);
        digitalWrite(Azul,0);
    }
}

```

```

}
if(lectura >= 25) // Temperatura mayor que 25°C: Color rojo(Calor)
{
    digitalWrite(Rojo,255);
    digitalWrite(Verde,0);
    digitalWrite(Azul,0);
}
}

```

### Explicación del Código

Se crean las diferentes variables que se van a utilizar en el proyecto, además de la invocación de las librerías:

- **ESP8266WiFi.h** :proporciona las rutinas específicas WiFi de ESP8266 a la que llamamos para conectarse a la red.
- **PubSubClient.h** :permite que nuestra placa se comporte como un cliente MQTT.
- **DHTesp.h** :permite obtener los valores de temperatura y humedad.

### Datos sobre la conexión a internet

En la etapa de conexión a internet se agrega el nombre del router, contraseña y la dirección IP del servidor MQTT. De ello, declaramos un objeto de clase *WiFiClient* , que permitirá establecer una conexión a una IP y puerto específicos. También declaramos un objeto de clase *PubSubClient* , que recibe como entrada el *WiFiClient* previamente definido.

### Función void setup()

En esta etapa inicializamos los pines de los sensores que vamos a utilizar, también llamaremos a la función *inicializando\_wifi()* que nos permitirá conectar a una red WiFi.

Una vez establecido el cliente en la etapa anterior se procederá a crear una conexión, para esto usaremos el método *setServer* que requiere un servidor y un puerto de intermediario. Ahora si queremos recibir mensajes, también debemos crear una función de devolución de llamada y configurarla usando el método *setCallback*.

## Función void loop()

En la función loop se comprobará la conexión usando el método *connect* llamando en *PubSubClient*, este devolverá verdadero si se establece la conexión o falso de lo contrario.

Será necesario llamar al método loop de *PubSubClient* para permitir que el cliente procese los mensajes entrantes y mantenga su conexión con el servidor. En esta etapa también se realizará la lectura de los sensores DHT22 y el sensor de humedad del suelo. Los resultados que se obtienen se mostrarán en el monitor serial, estos luego serán publicados en el NODE-RED.

La función *enciendeBomba* permitirá encender la bomba durante 15 segundos si el sensor de humedad del suelo supera el valor de 700.

La función *ledRGB* será necesaria para indicar la temperatura que el sensor DHT22 no dé. Si la temperatura es menor a 28°C se encenderá un led de color azul, representando a “Frio”. Entre 18 y 25 °C se encenderá un led de color verde, representando una temperatura “Ambiente” y si es mayor o igual a 25°C, se encenderá el led de color rojo, representando “Calor”.

## Conexión a Wifi

Para conectar el ESP8266 a una red Wifi, se debe conocer su SSID y contraseña. Lo configuramos como modo estación para que el ESP8266 se conecte a un punto de acceso. La conexión a una red Wi-Fi puede llevar un tiempo, por lo que generalmente agregamos un ciclo while en el que se verifica el estado de la conexión mediante el uso del *WiFi.status()*.

Cuando el ESP8266 está configurado como una estación Wi-Fi, puede conectarse a otras redes (como su enrutador). En este escenario, el enrutador asigna una dirección IP única a su placa ESP8266. Para obtener la dirección IP de su tablero, debe llamar *WiFi.localIP()* después de establecer una conexión con su red. El *randomSeed(micros())* inicializará un generador de números pseudoaleatorios, haciendo que se inicie en un punto arbitrario en una

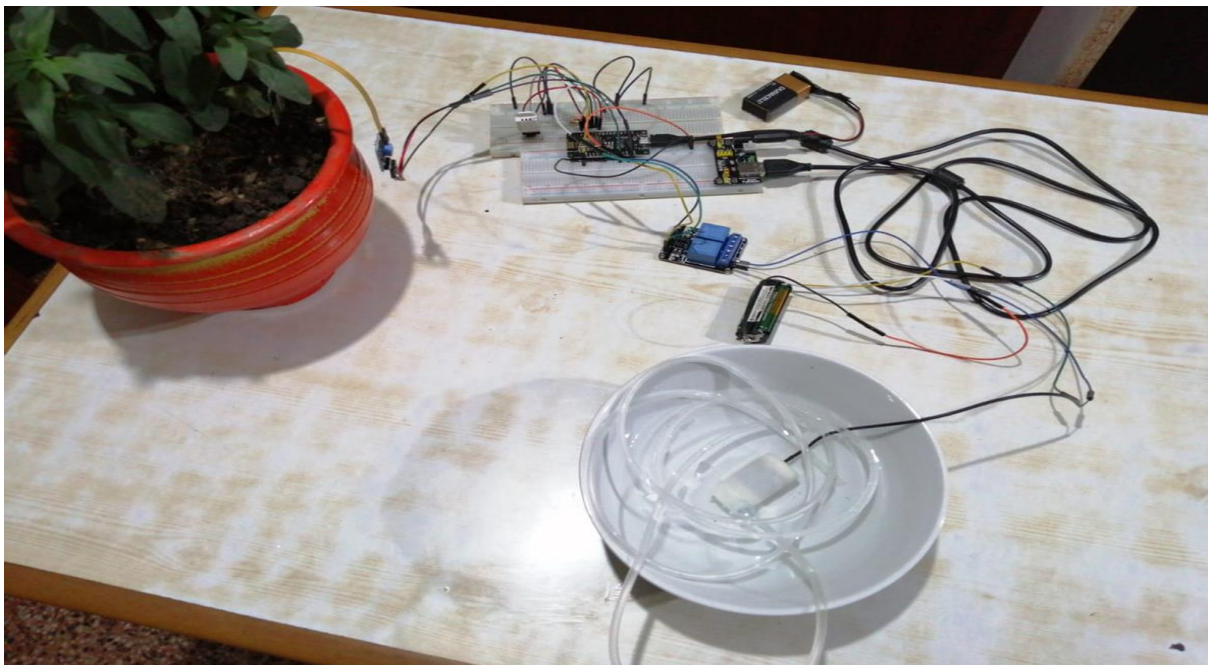
secuencia aleatoria. Como parámetro le enviaremos la función `micros()` que devolverá el número de microsegundos desde que la placa Arduino empezó a ejecutar el programa actual.

### Conexión MQTT

En esta parte evaluaremos si el cliente está conectado mediante la función `connected`. La función devolverá verdadero si el cliente está conectado, falso en caso contrario.

Si no está conectado utilizaremos la función `connect` donde enviaremos el nombre del dispositivo que estamos utilizando. Esta función devolverá verdadero si la conexión fue exitosa, si el resultado es falso se mostrará un mensaje de error, que con la función `state()` se podrá obtener más información sobre ello.

### Sistema de Riego Autónomo Implementado



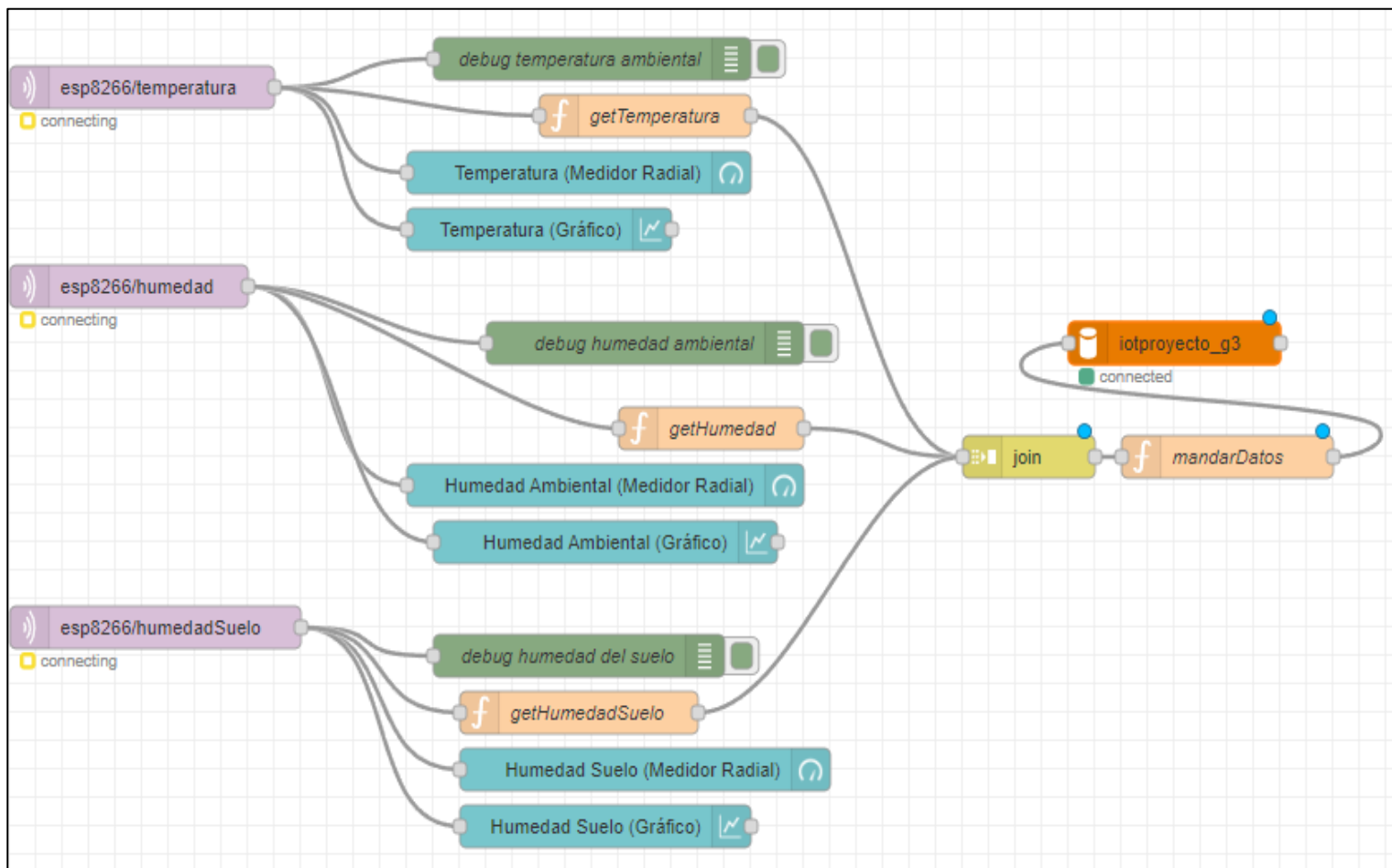
Se usó una placa ESP8266 conectada a los sensores DHT22 y el Sensor de humedad de suelo. Los datos recogidos de estos sensores serán enviados a través del protocolo MQTT a NODE-RED. Para accionar y conectar la bomba de agua se usó un relé de 12V, con el fin de proporcionar energía a la bomba para que realice su función. Además, está conectada a una batería de 9V para la alimentación apoyándose en una placa para alimentación de protoboard.

## Esquema de NODE-RED

Se implementaron tres tópicos al cual se suscribió el sistema para el envío de datos, los cuales son:

- *esp8266/temperatura*
- *esp8266/humedad*
- *esp8566/humedadSuelo*

Se enviará los datos de los sensores para poder ser presentados en los dashboard correspondientes. Además, se almacenan dichos datos en un nodo de tipo *JOIN*, en el cual se recolecta información de los tópicos para ser enviados a una función encargada de enviar los datos a una base de datos MySQL para su almacenamiento.



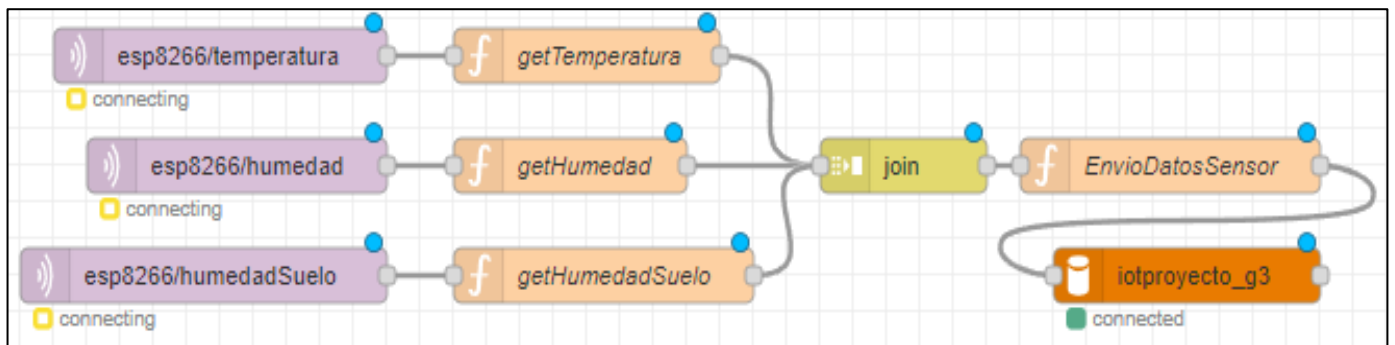
Esquema NODE-RED implementado



## Resultados del Proyecto

En base a los mensajes obtenidos de parte del MQTT broker, en la página de despliegue del Node red se pueden apreciar los gráficos a partir del flujo de nodos detallados en el esquema definido. Estos nodos gráficos han sido divididos en 3 grupos principales definidos por los valores registrados de los sensores conectados: Temperatura Ambiental, Humedad Ambiental y Humedad del suelo. En cuanto a la base de datos, se enviará la instrucción mediante el siguiente flujo:

- Se suscribe al *Broker MQTT*, en cada uno de los tres tópicos.
- Los mensajes enviados son recibidos por los 3 *getters*
- Se almacenan en un array, los mensajes mediante el nodo *JOIN*
- Se envía la sentencia SQL con los datos de la fecha y hora de envío en un tópico como se puede visualizar:



Properties

Name EnvioDatosSensor

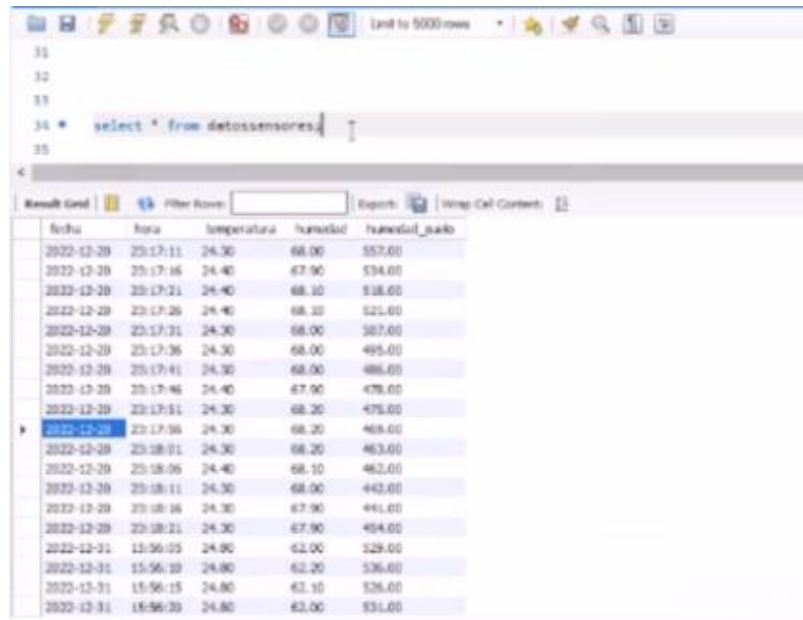
Setup On Start On Message On Stop

```
1 var temperatura = msg.payload[0];
2 var humedad = msg.payload[1];
3 var humedadSuelo = msg.payload[2];
4 msg.topic = "insert into datossensores values(CURDATE(), CUR
5 return msg;
```

### ***Función de Envío de Datos:***

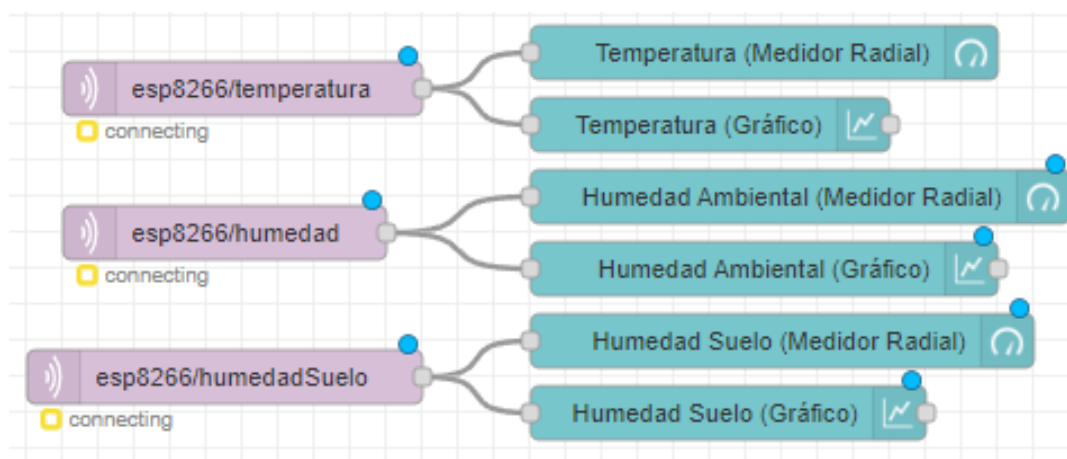
```
var temperatura = msg.payload[0];
var humedad = msg.payload[1];
var humedadSuelo = msg.payload[2];
msg.topic = "insert into datossensores values(CURDATE(), CURTIME()," +
temperatura + "," + humedad + "," + humedadSuelo+");";
return msg;
```

Los resultados obtenidos a partir de las inserciones en la base de datos MYSQL son los siguientes:



fecha	hora	temperatura	humedad	humedad_suelo
2022-12-29	23:17:11	24.30	66.00	557.00
2022-12-29	23:17:16	24.40	67.90	534.00
2022-12-29	23:17:21	24.40	68.10	518.00
2022-12-29	23:17:26	24.40	68.10	521.00
2022-12-29	23:17:31	24.30	68.00	507.00
2022-12-29	23:17:36	24.30	68.00	495.00
2022-12-29	23:17:41	24.30	68.00	486.00
2022-12-29	23:17:46	24.40	67.90	478.00
2022-12-29	23:17:51	24.30	68.20	475.00
2022-12-29	23:17:56	24.30	68.20	468.00
2022-12-29	23:18:01	24.30	68.20	463.00
2022-12-29	23:18:06	24.40	68.10	463.00
2022-12-29	23:18:11	24.30	68.00	442.00
2022-12-29	23:18:16	24.30	67.90	441.00
2022-12-29	23:18:21	24.30	67.90	404.00
2022-12-31	15:56:05	24.80	62.00	529.00
2022-12-31	15:56:10	24.80	62.20	536.00
2022-12-31	15:56:15	24.80	62.10	536.00
2022-12-31	15:56:20	24.80	62.00	531.00

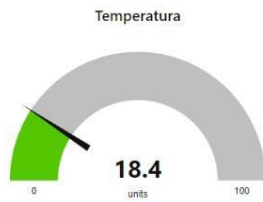
Estos serán representados en dos tipos de diagramas. Un medidor radial y un gráfico de seguimiento. Con el que se puede observar, tanto el rango de valores del sensor analógico de 0 a 1024 en el caso del sensor de Humedad y el DHT devuelve en porcentaje.



*Representación del Dashboard de Node Red*

## Tablero de control

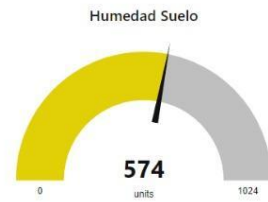
### Temperatura Ambiental



### Humedad Ambiental



### Humedad Suelo



*Dashboard Resultado obtenido de la ejecución del Node-RED*

## Conclusiones

- Con el fin enviar los datos tópicos definidos a la base de datos MYSQL (compatible con NODE-RED), se tuvo que formular la sentencia SQL a través de la unión de cadenas de texto en un solo mensaje, para poder insertar los valores adquiridos en la base de datos.
- En la implementación del sistema actuador de la bomba de agua, se estableció un tiempo de delay adicional necesario, para que no compruebe de manera continua. Como también se le agrego este delay en el proceso de apagado.
- En la construcción del código ESP8266, se tuvo en cuenta el protocolo MQTT, que nos permite el envío y recepción de mensajes a través de los tópicos implementados. Para el envío de valores se colocó la función “*payload*” del mensaje, que contendrá el cuerpo del mensaje.
- El sensor analógico de humedad del suelo entrega valores en el rango de 0 a 1024 como del DHT22 que da una representación de 0 a 100, tanto en temperatura como en humedad del ambiente, este valor es dependiente del sensor utilizado, en las cuales se necesiten un mapeo de valores para establecer ese rango descrito.

## Bibliografía

- Calle, Q. E. R. (2021, 14 octubre). *Repositorio Digital UCSG: Diseño de un sistema de riego automatizado mediante proteus, con tecnología arduino y variador de frecuencia aplicado a un motor asíncrono*.  
<http://repositorio.ucsg.edu.ec/handle/3317/17189>
- Castro-Silva, J. A. (2016, 31 mayo). *Sistema de riego autónomo basado en la Internet de las Cosas*. <https://reunir.unir.net/handle/123456789/3648>
- Anaya-Isaza, A., Peluffo-Ordoñez, D. H., Ivan-Rios, J., Castro-Silva, J. A., Ruiz, D. C., Llanos, L. E. (2016). Sistema de riego basado en la Internet de las cosas (IoT). *Conferencias Internacionales FICA*.
- M. Lekić & G. Gardašević, "IoT sensor integration to Node-RED platform," *2018 17th International Symposium INFOTEH-JAHORINA (INFOTEH)*, 2018, pp. 1-5, doi: 10.1109/INFOTEH.2018.8345544.
- Viera, Y. C., Borrego, J. M., & Viera, E. C. (2021). Propuesta de metodología para el diseño de dashboard. *Revista cubana de transformación digital*, 2(3), 56-76.