1 MANUAL DE USUARIO DE LA IMPLEMENTACIÓN DE SISTEMA DE CONTROL IOT

1.1 Configuración de la Raspberry Pi

- Preparación de la Raspberry Pi: Descarga la imagen del sistema operativo específicamente diseñado para funcionar con el sensor hidrómetro TILT desde el sitio web oficial TILT Hydrometer (https://tilthydrometer.com). Utiliza la herramienta Raspberry Pi Imager para grabar esta imagen en una tarjeta SD.
- Retira la tarjeta microSD de la computadora y conéctala nuevamente, accede a la partición "boot" de la tarjeta microSD, ejecuta el fichero "SETUP.html" y configura la red Wifi y establece una cuenta de correo recibir avisos. Guarda cualquier archivo generado durante este proceso en la raíz de la partición "boot".
- Inserta la tarjeta SD en tu Raspberry Pi y enciéndela.
- Una vez que el sistema operativo ha arrancado completamente, puedes acceder a la Raspberry Pi. Esto puede hacerse a través de una interfaz gráfica (si está disponible) o mediante una conexión SSH para administrar la Pi a través de la línea de comandos.
- Es recomendable ejecutar los comandos de actualización (sudo apt-get update y sudo apt-get upgrade)

1.2 Conexión de Sensores

Sensor DHT22:

- Conecte el sensor DHT22 al pin GPIO4 de la Raspberry Pi 4.

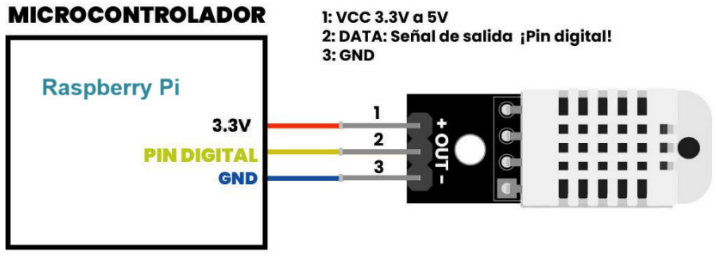

Figura 1.1 Conexión del sensor DHT22

- Instalación de la librería Adafruit_DHT

  sudo apt install python3-pip

  pip3 install Adafruit_DHT

- Configura un script en Python para capturar los valores de temperatura y humedad y almacenarlos en la base de datos MySQL, el código de implementación está detallado en el ANEXO A de este manual.

Sensor TILT:

- Enciende el Bluetooth en la Raspberry Pi.
- Busca el sensor TILT a través del Bluetooth de la Raspberry Pi y empareja el dispositivo.
- También puedes emparejar el sensor TILT con tu dispositivo móvil. Descarga e instala la aplicación desde la Play Store para visualizar los datos del sensor. Asegúrate de mantener una conexión directa entre el móvil y el sensor para acceder a la información en tiempo real.

Sensor PH:

- Para el sensor PH es analógico se necesita utilizar un Arduino y conectarlo conforme se indica en el siguiente gráfico.

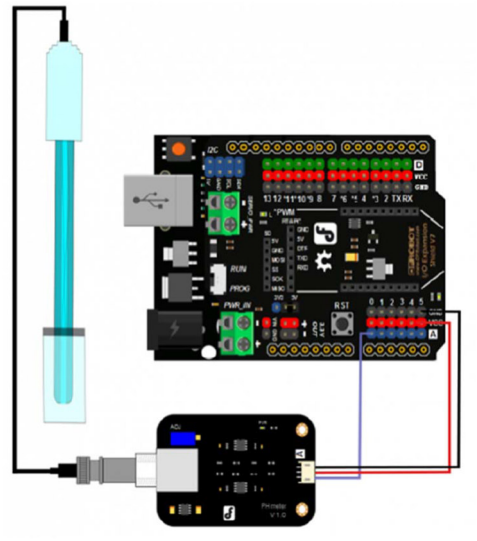

Figura 1.2 Diagrama de conexión del Arduino a PH

- Configura un script en Python para capturar los valores de PH enviados desde el Arduino hacia el raspberry pi y almacenarlos en la base de datos MySQL, el código de implementación está detallado en el ANEXO A de este manual.
- Al primer uso se recomienda calibrar el PH por software conforme lo indica el fabricante.

Para calibrar el pH utilizando el medidor de pH (SEN0161), puedes seguir estos pasos:

Método de Calibración de Software

Conexión: Conecta el electrodo de pH al conector BNC en la placa del medidor de pH y conecta la placa al puerto analógico 0 del controlador Arduino.

Sube el Código: Carga el código que se detalla en el ANEXO B en el controlador Arduino.

Preparación: Abre el monitor serial en el IDE de Arduino y asegúrate de que esté configurado para "Both NL & CR" y a 115200 baudios.

Calibración Ácida: Lava el electrodo de pH con agua pura (agua destilada es lo mejor) y sécalo. Inserta el electrodo en una solución estándar ácida con pH = 4.0. Espera a que las lecturas se estabilicen.

Escribe "acid:4.00" en el monitor serial y presiona Enter. Deberías ver un mensaje que dice "Calibración Ácida Exitosa".

Calibración Alcalina: Limpia el electrodo nuevamente y luego insértalo en una solución estándar alcalina con pH = 10.0. Espera a que las lecturas se estabilicen.

Escribe "alkali:10.00" en el monitor serial y presiona Enter. Deberías ver un mensaje que dice "Calibración Alcalina Exitosa".

Finaliza la Calibración: Escribe "exit" para salir del modo de calibración. Verás un mensaje que dice "Calibración Exitosa, Salir del Modo de Calibración".

Verificación: Comprueba si el medidor de pH fue calibrado correctamente utilizando soluciones con pH = 4.00, 9.18 y 10.00. Las lecturas deben estar dentro de un error de 0.1.

1.3 Configuración del Servidor en AWS

Instanciación de Amazon EC2:

Para instanciar una máquina virtual en Amazon EC2, sigue estos pasos:

- Accede a la AWS Management Console.
- Inicia sesión con tus credenciales de AWS.
- En el menú de servicios, selecciona EC2 para acceder al panel de administración de instancias.
- Haz clic en Launch Instance para iniciar el proceso de creación de una nueva instancia.
- Selecciona una imagen de Ubuntu Server 20.04 LTS (HVM) ,SSD Volume Type**.**
- En el paso de Key Pair, selecciona Create a new key pair, ingresa un nombre para el par de claves.
- Descarga el archivo .pem generado al hacer clic en Download Key Pair. Guarda este archivo en un lugar seguro.
- Una vez que la instancia esté en estado "Running", selecciona la instancia y haz clic en **Connect** para obtener instrucciones sobre cómo conectarte mediante SSH.

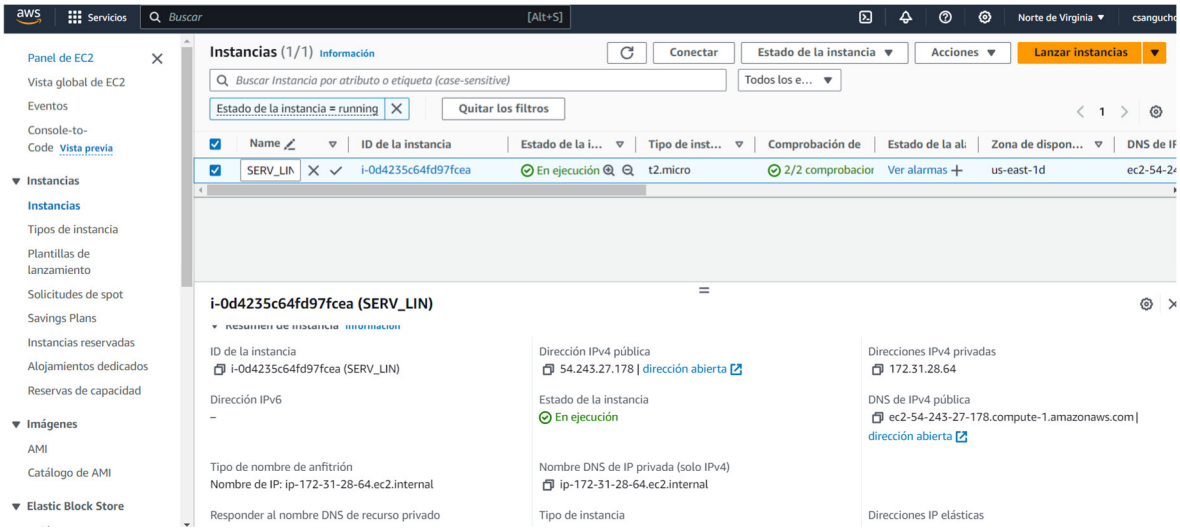

Figura 1.3 Instancia creada en AWS

Acceso SSH con PUTTY:

Para acceso con PuTTY, sigue estos pasos:

- Descarga, instala y ejecuta PuTTYgen, ve a Conversions y selecciona Import Key. Selecciona el archivo .pem que descargaste y haz clic en Open. Luego, guarda la clave privada en formato .ppk haciendo clic en Save private key.
- Abrir PuTTY, configurar la Conexión SSH, Ve a Connection -> SSH -> Auth->Credentials. En Private key file for authentication, haz clic en Browse y selecciona el archivo .ppk que guardaste.
- En la categoría Session, ingresa la dirección IP pública de tu instancia EC2 y conectar.
- Actualiza los paquetes de software e instala el paquete net-tools.

  apt-get update

  apt-get upgrade

  apt-get install net-tools

Instalación de Node-RED:

Procede a instalar **Node.js** y **Node-RED**:

```
curl -sL https://deb.nodesource.com/setup_12.x | sudo -E bash -

sudo apt-get install -y nodejs build-essential

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

Configuración de Reglas de Entrada: En el grupo de seguridad de la instancia EC2, configure una regla de entrada para permitir tráfico TCP en el puerto 1880 desde cualquier IP (0.0.0.0/0).

Inicie el servicio ejecutando node-red y acceda a la interfaz web en http://[IP_DEL_SERVIDOR]:1880/.

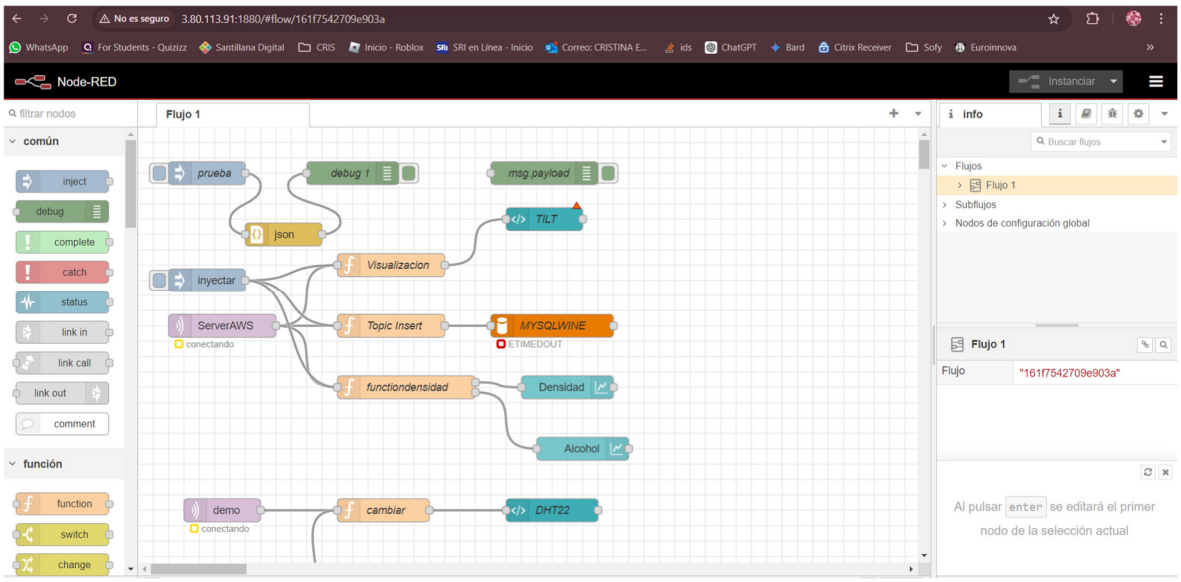

Figura 1.4 Interfaz web Node-red

1.4 Configuración de Base de Datos

Instalación de MariaDB:

- Instalar el servidor de base de datos MariaDB con el siguiente comando:

  sudo apt-get install mariadb-server

- Inicia sesión en el servidor MySQL utilizando el usuario root:

  mysql -u root

- Crea un nuevo usuario llamado xxx y otórgale todos los privilegios:

  CREATE USER xxx@'%' IDENTIFIED BY yyy.';

  GRANT ALL PRIVILEGES ON *.* TO 'xxx'@'%';

- Modifica la configuración para permitir conexiones remotas:

  cd /etc/mysql/mariadb.conf.d/

  sudo nano 50-server.cnf

  Cambia la línea bind-address de 127.0.0.1 a 0.0.0.0.bash

- Luego reinicia el servicio de mysql

  sudo systemctl stop mysql

```
sudo systemctl start mysql
```

Instalación de DBeaver:

Instala DBeaver, es una herramienta de administración de bases de datos, y crea una nueva conexión usando las credenciales del usuario creado de MariaDB.

Crea una base de datos denominada BDDWINE y luego define dos tablas para almacenar los datos:

```
CREATE TABLE BDDWINE.ReadDHT22 (
   ID_DHT22 INT auto_increment NOT NULL PRIMARY KEY,
   timestamp TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
   Atemperatura FLOAT(5,2),
   humedad FLOAT(5,2)
)
CREATE TABLE BDDWINE.ReadTILT (
   ID INT auto_increment NOT NULL PRIMARY KEY,
   timestamp TEXT,
   temp FLOAT,
   densidad FLOAT,
   brix FLOAT,
   alcohol FLOAT
)
```

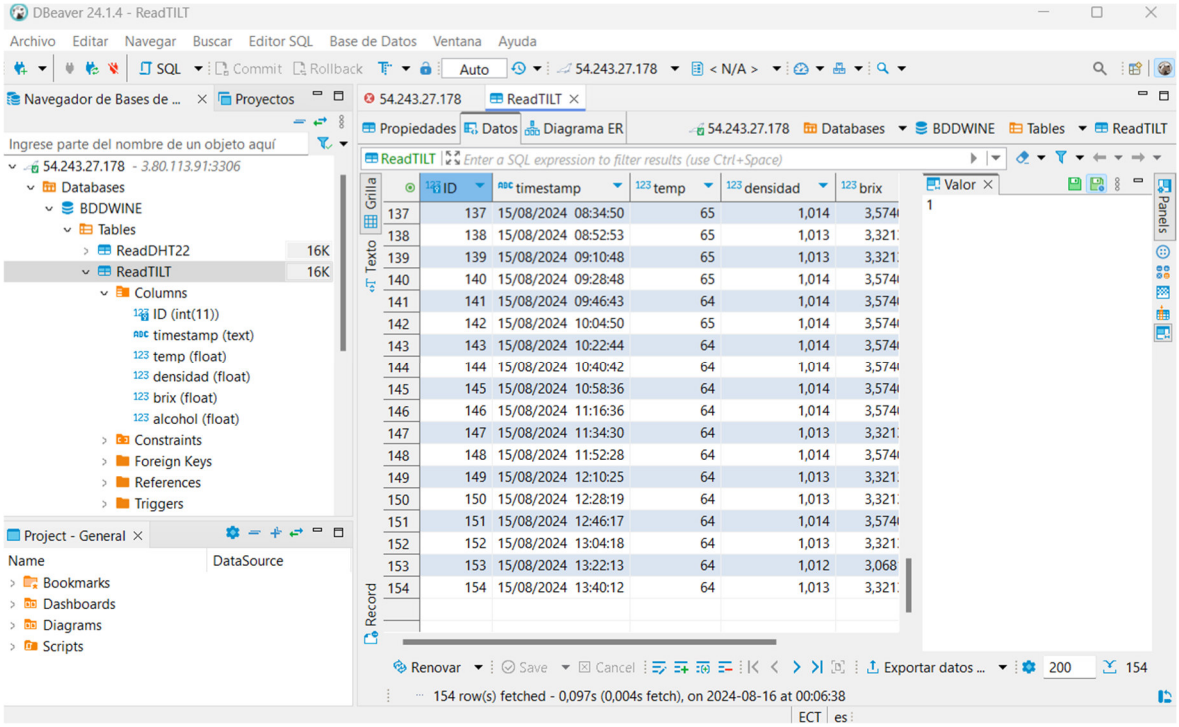

Figura 1.5 Herramienta de administración de bases de datos DBeaver

Conexión de Raspberry Pi a la base de datos:

Para enviar información a una base de datos desde una Raspberry Pi se usa la librería mysql-connector-python. A continuación, se describe el proceso para instalar y configurar esta librería en tu Raspberry Pi 3:

```
pip3 install mysql-connector-python
```

Crear un script en Python para establecer una conexión a la base de datos BDDWINE, un ejemplo se muestra a continuación:

```
# Importar la librería mysql.connector

import mysql.connector as mysql

# Establecer la conexión con la base de datos MySQL

db = mysql.connect(

    host=" 3.80.113.91",    # Dirección IP del servidor MySQL

    user="usuario",         # Nombre de usuario para acceder a MySQL

    passwd="contraseña",   # Contraseña del usuario de MySQL
```

```
    database=" BDDWINE "   # Nombre de la base de datos a la que conectarse
)
# Crear un cursor para ejecutar consultas
cursor = db.cursor()
# Ejecutar una consulta SQL
cursor.execute(query,valores)
# Confirmar los cambios en la base de datos
db.commit()
cursor.close()
db.close()
print("Fin de programa")
```

1.4.1.1 Configuración de MQTT

Configuración de MQTT en Raspberry Pi:

- Instalar mosquitto en el raspberry

  sudo apt-get install mosquitto

- Instalar clientes MQTT en Raspberry Pi

  sudo apt-get install mosquitto-clients

- Instalar la librería paho-mqtt:

  sudo pip3 install paho-mqtt

  pip install --upgrade paho-mqtt

- Crear un archivo de contraseñas y agregar un usuario:

  Sudo mosquitto_passwd -c /etc/mosquitto/passwd usuario1

- Editar el archivo de configuración de Mosquitto

  sudo nano /etc/mosquitto/mosquitto.conf

- Añadir o modificar las siguientes líneas

```
password_file /etc/mosquitto/passwd

allow_anonymous false
```

- Reiniciar el servicio de Mosquitto para aplicar los cambios.

```
sudo systemctl restart mosquitto
```

Crear un script en Python para establecer una conexión al Broker MQTT y publicar un mensaje, un ejemplo se muestra a continuación:

```
import paho.mqtt.client as mqtt

cliente = mqtt.Client()

cliente.username_pw_set(username="usuario1",password="contraseña")

cliente.connect("3.80.113.91 ",1883)

data = "Hola soy Python"

cliente.publish("demo",data)

print("Fin de programa")
```

Para permitir que el tráfico MQTT pase a través del puerto 1883, agregar una regla de entrada al servidor AWS.

Configuración de MQTT en Servidor AWS:

Para configurar un broker MQTT en un servidor, sigue estos pasos:

- Instalar Mosquitto en tu servidor

```
sudo apt-get install mosquitto mosquitto-clients
```

- Configurar la autenticación en Mosquitto, genera un archivo de contraseñas y configurar el archivo de configuración.

Configuración de MQTT en Node-red:

- Accede a tu interfaz de Node-RED a través del navegador.

- En la paleta de nodos de Node-RED, busca el nodo llamado **"mqtt in"** o **"mqtt out"**. Arrástralo al flujo de trabajo.
- Hacer doble clic en el nodo MQTT para abrir su configuración, y configurar conforme se indica en el siguiente gráfico.

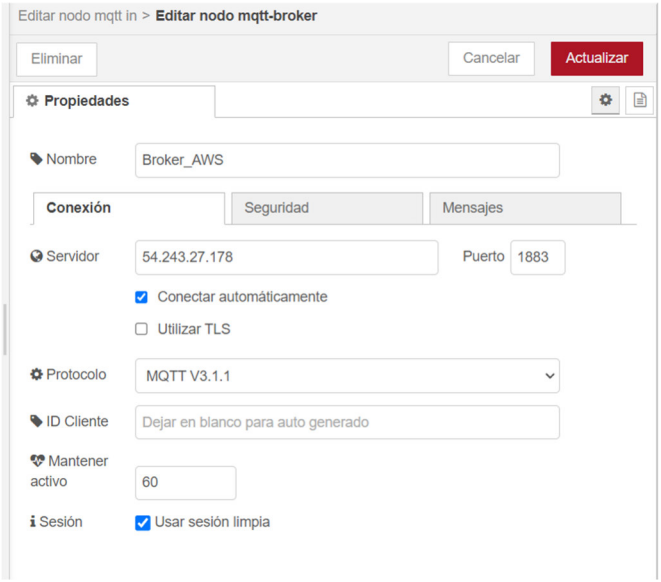

Figura 1.6 Configuración de MQTT con seguridad

1.5 Integración de TILT Hidrómetro con MQTT

Modificar el flujo de Node-red en la Raspberry Pi y publicar los mensajes del sensor a través de MQTT, sigue estos pasos:

- En el editor de Node-RED, arrastra un nodo "mqtt out" desde la paleta de nodos al área de trabajo.
- Conecta el nodo que envía los datos del sensor al nodo **"mqtt out"**.
- Haz clic en el botón **"Deploy"** en la parte superior derecha para aplicar los cambios y activar el flujo.
- Para verificar el formato JSON e los mensajes que circulan, ingresa en el navegador la siguiente URL:

  http://tiltpi.local:1880/data/RED.json

Figura 1.7 Formato del mensaje JSON

Para procesar y almacenar los datos recibidos a través de MQTT en la base de datos, debes seguir estos pasos en Node-RED:

- Agregar un nodo **"function"** en el flujo para transformar el mensaje en texto. Utiliza el siguiente código:

```
// La cadena de texto viene en msg.payload

var data = msg.payload.split(",");

// Asignar variables

var v1 = data[0];

var v2 = data[2];

var v3 = data[3];

var v4 = data[4];

var v5 = data[9];

msg.topic = "INSERT INTO ReadTILT(timestamp,temp,densidad,brix,alcohol) VALUES('"+v1+"','"+v2+"','"+v3+"','"+v4+"','"+v5+"')"

return msg;
```

  El campo msg.topic en el nodo function contendrá la consulta SQL que debe ejecutarse.

- Agregar un nodo mysql en el flujo y configura los datos de conexión a tu base de datos. La salida del nodo funtion debe conectarse a la entrada del nodo mysql.

- Para validad el contenido del mensaje de salida, conectar el nodo **"debug".** Esto nodo es útil para mostrar información de depuración.

1.6 Dashboard node-red del Servidor AWS

Para visualizar los datos del Sensor TILT en el panel de control seguir estos pasos:

- Agregar un nodo function para convertir el mensaje de texto en un formato JSON. El código para este cambio es el siguiente:

```
var data = msg.payload.split(",");

// Asignar variables

var v1 = data[0];  // timestamp

var v2 = data[2];  // temperatura

var v3 = data[3];  // densidad

var v4 = data[4];  // brix

var v5 = data[9];  // alcohol

// Crear un nuevo payload estructurado como un objeto JSON

msg.payload = {

   timestamp: v1,

   temperatura: v2,

   densidad: v3,

   brix: v4,

   alcohol: v5

};

return msg;
```

- Conectar la Salida del Nodo function al Nodo template:
- Añade un nodo template, crea un grupo denominado "Control y Monitoreo" en la pestaña "Inicio". El código visualizar para mostrar los datos en el Dashboard. se detalla en el ANEXO B.

Para visualizar los datos de forma gráfica, sigue estos pasos:

- Agregar el Nodo de Función para preparar los datos para el gráfico. Utiliza el siguiente código para asignar los datos a dos mensajes separados una para

"Densidad" y otra para "Alcohol". Asegurarse que se configure en la pestaña de configuración las Salidas en 2.

```
var data = msg.payload.split(",");

// Asignar variables

var densidad = data[3];  // Posición correcta para "densidad"

var alcohol = data[9];   // Posición correcta para "alcohol"

// Crear un nuevo objeto para la segunda salida

var msg1 = {};

// Asignar valores al payload y topic de cada mensaje

msg.payload = densidad*1000;

msg.topic = "Densidad";

msg1.payload = alcohol;

msg1.topic = "Alcohol";

// Retornar los dos mensajes en un array

return [msg, msg1];
```

- Arrastra dos nodos chart (nodo de gráfico) desde la paleta de nodos al área de trabajo. Cada uno de estos nodos representará un gráfico diferente.
- Conecta la primera salida del nodo de función al primer nodo chart, que se encargará de visualizar la Densidad.
- Conecta la segunda salida del nodo de función al segundo nodo chart, que mostrará el Alcohol.
- Añade los nodos chart al grupo denominado Gráficas en el Dashboard de Node-RED. Esto organiza los gráficos de manera que se muestren en el panel de control bajo una sección común.
- Haz clic en el botón Deploy para aplicar los cambios y activar los flujos.

ANEXO A

```
import serial
import re
import mysql.connector as mysql
import time
import paho.mqtt.client as mqtt
import json
import Adafruit_DHT as dht
serial_sensor = serial.Serial("/dev/ttyACM0", 115200)
dht_sensor = dht.DHT22
db = mysql.connect(host="98.81.220.163", user="cristy", passwd="Liz4040.", database="BDDWINE")
cursor = db.cursor()
query_ph = 'INSERT INTO ReadPH(PH) VALUES (%s)'
query_dht = 'INSERT INTO ReadDHT22(Atemperatura, humedad) VALUES (%s, %s)'
cliente_mqtt = mqtt.Client()
cliente_mqtt.username_pw_set(username="tesis", password="Liz4040.")
cliente_mqtt.connect("98.81.220.163", 1883)
try:
    while True:
        dato = serial_sensor.readline().decode().strip()
        match = re.search(r"pH:\s*([0-9]*\.?[0-9]+)", dato, re.IGNORECASE)
        if match:
            PH = round(float(match.group(1)), 2)
            print(f"pH = {PH}")
```

```
    cursor.execute(query_ph, (PH,))
    db.commit()
    print("Lectura de pH insertada en la base de datos")
    datos_dict = {'PH': PH}
    datos_json = json.dumps(datos_dict)
    cliente_mqtt.publish("readPH", datos_json)
    print("Datos de pH publicados en MQTT")
else:
    print("Error al leer el sensor pH")
humidity, temperature = dht.read_retry(dht_sensor, 4)
if humidity is not None and temperature is not None:
    temperature = round(temperature, 2)
    humidity = round(humidity, 2)
    print(f"Temperatura = {temperature} oC, Humedad = {humidity} %")
    cursor.execute(query_dht, (temperature, humidity))
    db.commit()
    print("Lectura de DHT22 insertada en la base de datos")
    datos_dict = {
        'temperature': temperature,
        'humidity': humidity
    }
    datos_json = json.dumps(datos_dict)
    cliente_mqtt.publish("readDHT22", datos_json)
    print("Datos de DHT22 publicados en MQTT")
```

```
        else:
            print("Error al leer el sensor DHT22")
        time.sleep(14400)
except KeyboardInterrupt:
    print("Fin de programa")
```

ANEXO B

```
#include <EEPROM.h>

#define EEPROM_write(address, p) {int i = 0; byte *pp = (byte*)&(p); for(; i
< sizeof(p); i++) EEPROM.write(address + i, pp[i]);}
#define EEPROM_read(address, p) {int i = 0; byte *pp = (byte*)&(p); for(; i
< sizeof(p); i++) pp[i] = EEPROM.read(address + i);}

#define ReceivedBufferLength 20
char receivedBuffer[ReceivedBufferLength + 1];
byte receivedBufferIndex = 0;

#define SCOUNT 30
int analogBuffer[SCOUNT];
int analogBufferIndex = 0;

#define SlopeValueAddress 0
#define InterceptValueAddress (SlopeValueAddress + 4)
float slopeValue, interceptValue, averageVoltage;
boolean enterCalibrationFlag = 0;

#define SensorPin A0
#define VREF 5000

void setup() {
  Serial.begin(115200);
  readCharacteristicValues(); // read the slope and intercept of the pH
probe
}

void loop() {
  if (serialDataAvailable()) {
    byte modeIndex = uartParse();
```

```
    phCalibration(modeIndex);
    EEPROM_read(SlopeValueAddress, slopeValue);
    EEPROM_read(InterceptValueAddress, interceptValue);
  }

  static unsigned long sampleTimepoint = millis();
  if (millis() - sampleTimepoint > 40U) { // Enviar datos cada 30 minutos
    sampleTimepoint = millis();
    analogBuffer[analogBufferIndex] = analogRead(SensorPin) / 1024.0 * VREF;
    analogBufferIndex++;
    if (analogBufferIndex == SCOUNT)
      analogBufferIndex = 0;
    averageVoltage = getMedianNum(analogBuffer, SCOUNT);
  }

  static unsigned long printTimepoint = millis();
  if (millis() - printTimepoint > 1000U) {
    printTimepoint = millis();
    if (enterCalibrationFlag) {
      Serial.print("Voltage: ");
      Serial.print(averageVoltage);
      Serial.println("mV");
    } else {
      Serial.print("pH: ");
      Serial.println(averageVoltage / 1000.0 * slopeValue + interceptValue);
    }
  }
}

boolean serialDataAvailable(void) {
  char receivedChar;
  static unsigned long receivedTimeOut = millis();
  while (Serial.available() > 0) {
    if (millis() - receivedTimeOut > 1000U) {
      receivedBufferIndex = 0;
      memset(receivedBuffer, 0, (ReceivedBufferLength + 1));
    }
    receivedTimeOut = millis();
    receivedChar = Serial.read();
    if (receivedChar == '\n' || receivedBufferIndex == ReceivedBufferLength)
{
      receivedBufferIndex = 0;
      strupr(receivedBuffer);
      return true;
    } else {
```

```
      receivedBuffer[receivedBufferIndex] = receivedChar;
      receivedBufferIndex++;
    }
  }
  return false;
}

byte uartParse() {
  byte modeIndex = 0;
  if (strstr(receivedBuffer, "CALIBRATION") != NULL)
    modeIndex = 1;
  else if (strstr(receivedBuffer, "EXIT") != NULL)
    modeIndex = 4;
  else if (strstr(receivedBuffer, "ACID:") != NULL)
    modeIndex = 2;
  else if (strstr(receivedBuffer, "ALKALI:") != NULL)
    modeIndex = 3;
  return modeIndex;
}

void phCalibration(byte mode) {
  char *receivedBufferPtr;
  static byte acidCalibrationFinish = 0, alkaliCalibrationFinish = 0;
  static float acidValue, alkaliValue;
  static float acidVoltage, alkaliVoltage;
  float acidValueTemp, alkaliValueTemp, newSlopeValue, newInterceptValue;

  switch (mode) {
    case 0:
      if (enterCalibrationFlag)
        Serial.println(F("Command Error"));
      break;
    case 1:
      receivedBufferPtr = strstr(receivedBuffer, "CALIBRATION");
      enterCalibrationFlag = 1;
      acidCalibrationFinish = 0;
      alkaliCalibrationFinish = 0;
      Serial.println(F("Enter Calibration Mode"));
      break;
    case 2:
      if (enterCalibrationFlag) {
        receivedBufferPtr = strstr(receivedBuffer, "ACID:");
        receivedBufferPtr += strlen("ACID:");
        acidValueTemp = strtod(receivedBufferPtr, NULL);
        if ((acidValueTemp > 3) && (acidValueTemp < 5)) {
```

```
          acidValue = acidValueTemp;
          acidVoltage = averageVoltage / 1000.0;
          acidCalibrationFinish = 1;
          Serial.println(F("Acid Calibration Successful"));
        } else {
          acidCalibrationFinish = 0;
          Serial.println(F("Acid Value Error"));
        }
      }
      break;
    case 3:
      if (enterCalibrationFlag) {
        receivedBufferPtr = strstr(receivedBuffer, "ALKALI:");
        receivedBufferPtr += strlen("ALKALI:");
        alkaliValueTemp = strtod(receivedBufferPtr, NULL);
        if ((alkaliValueTemp > 8) && (alkaliValueTemp < 11)) {
          alkaliValue = alkaliValueTemp;
          alkaliVoltage = averageVoltage / 1000.0;
          alkaliCalibrationFinish = 1;
          Serial.println(F("Alkali Calibration Successful"));
        } else {
          alkaliCalibrationFinish = 0;
          Serial.println(F("Alkali Value Error"));
        }
      }
      break;
    case 4:
      if (enterCalibrationFlag) {
        if (acidCalibrationFinish && alkaliCalibrationFinish) {
          newSlopeValue = (acidValue - alkaliValue) / (acidVoltage -
alkaliVoltage);
          EEPROM_write(SlopeValueAddress, newSlopeValue);
          newInterceptValue = acidValue - (newSlopeValue * acidVoltage);
          EEPROM_write(InterceptValueAddress, newInterceptValue);
          Serial.print(F("Calibration Successful"));
        } else {
          Serial.print(F("Calibration Failed"));
        }
        Serial.println(F(",Exit Calibration Mode"));
        acidCalibrationFinish = 0;
        alkaliCalibrationFinish = 0;
        enterCalibrationFlag = 0;
      }
      break;
  }
```

```
}

int getMedianNum(int bArray[], int iFilterLen) {
  int bTab[iFilterLen];
  for (byte i = 0; i < iFilterLen; i++) {
    bTab[i] = bArray[i];
  }
  int i, j, bTemp;
  for (j = 0; j < iFilterLen - 1; j++) {
    for (i = 0; i < iFilterLen - j - 1; i++) {
      if (bTab[i] > bTab[i + 1]) {
        bTemp = bTab[i];
        bTab[i] = bTab[i + 1];
        bTab[i + 1] = bTemp;
      }
    }
  }
  if ((iFilterLen & 1) > 0)
    bTemp = bTab[(iFilterLen - 1) / 2];
  else
    bTemp = (bTab[iFilterLen / 2] + bTab[iFilterLen / 2 - 1]) / 2;
  return bTemp;
}

void readCharacteristicValues() {
  EEPROM_read(SlopeValueAddress, slopeValue);
  EEPROM_read(InterceptValueAddress, interceptValue);
  if (EEPROM.read(SlopeValueAddress) == 0xFF &&
EEPROM.read(SlopeValueAddress + 1) == 0xFF && EEPROM.read(SlopeValueAddress
+ 2) == 0xFF && EEPROM.read(SlopeValueAddress + 3) == 0xFF)
  {
    slopeValue = 3.5;
    EEPROM_write(SlopeValueAddress, slopeValue);
  }
  if (EEPROM.read(InterceptValueAddress) == 0xFF &&
EEPROM.read(InterceptValueAddress + 1) == 0xFF &&
EEPROM.read(InterceptValueAddress + 2) == 0xFF &&
EEPROM.read(InterceptValueAddress + 3) == 0xFF)
  {
    interceptValue = 0;
    EEPROM_write(InterceptValueAddress, interceptValue);
  }
}
```