**UNIVERSIDAD NACIONAL MAYOR DE SAN MARCOS**

**Universidad del Perú, Decana de América**

**FACULTAD DE INGENIERÍA DE SISTEMAS E INFORMÁTICA**

**Escuela Profesional de Ingeniería de Sistemas**



**Sistema de Riego Autónomo**

**Grupo 1**

**Presentado por**

Agurto Briceño Erick Joel – 19200001

Arca Barra Valerie Jazmin – 19200191

Cahuana Guerra Leonardo Sergio – 19200234

Guillen Melgarejo Gabriel Omar – 19200019

Puyen Huapaya Jafet Raul – 19200039

Reynoso Armas Jhonas Raul – 19200135

**Docente a cargo**

Rosas Cueva Yessica

**Asignatura**

Internet de las Cosas

**Agosto, 2022**

**Grupo 1**

Somos el grupo 1 del curso de Internet de las cosas, cuyos integrantes y porcentaje de participación son los siguientes:

|  |  |
| --- | --- |
| **Integrantes** | **Porcentaje de participación** |
| Agurto Briceño Erick Joel | 100% |
| Arca Barra Valerie Jazmin | 100% |
| Cahuana Guerra Leonardo Sergio | 100% |
| Guillen Melgarejo Gabriel Omar | 100% |
| Puyen Huapaya Jafet Raul | 100% |
| Reynoso Armas Jhonas Raul | 100% |

**INDICE**

[1. Introducción 1](#_Toc112630304)

[2. Planteamiento del estudio 2](#_Toc112630305)

[2.1. Planteamiento del problema 2](#_Toc112630306)

[2.1.1. Problema general 2](#_Toc112630307)

[2.1.2. Problemas específicos 2](#_Toc112630308)

[2.2. Objetivos 2](#_Toc112630309)

[2.2.1. Objetivo general 2](#_Toc112630310)

[2.2.2. Objetivos específicos 2](#_Toc112630311)

[3. Marco teórico 3](#_Toc112630312)

[3.1. Antecedentes 3](#_Toc112630313)

[4. Bases teóricas 5](#_Toc112630314)

[4.1. Riego 5](#_Toc112630315)

[4.2. Métodos de Riego 5](#_Toc112630316)

[4.2.1. Inundación y sumersión 5](#_Toc112630317)

[4.2.2. Riego por surcos 5](#_Toc112630318)

[4.2.3. Riego por goteo 6](#_Toc112630319)

[4.3. Sistema de automatización de riego 6](#_Toc112630320)

[4.3.1. Domótica. 7](#_Toc112630321)

[4.3.2. Características del sistema domótico. 7](#_Toc112630322)

[4.4. Componentes de un sistema de riego. 7](#_Toc112630323)

[4.4.1. Protoboard 7](#_Toc112630324)

[4.4.2. Led 8](#_Toc112630325)

[4.4.3. Módulo Relevador de 1 canal 8](#_Toc112630326)

[4.4.4. Bomba de agua 9](#_Toc112630327)

[4.4.5. ESP8266 10](#_Toc112630328)

[4.4.6. Placa de desarrollo NodeMCU 11](#_Toc112630329)

[4.4.7. Sensor de humedad de suelo 11](#_Toc112630330)

[4.4.8. Sensor DHT22 12](#_Toc112630331)

[4.4.9. Sensor de ultrasonido HC-SR04 12](#_Toc112630332)

[4.5. Protocolo de comunicación MQTT 14](#_Toc112630333)

[4.5.1. Arquitectura 14](#_Toc112630334)

[4.5.2. Broker 14](#_Toc112630335)

[4.5.3. Cliente 14](#_Toc112630336)

[4.6. Node – RED 15](#_Toc112630337)

[5. Componentes del sistema de riego 16](#_Toc112630338)

[6. Implementación del sistema 18](#_Toc112630339)

[6.1. Diagrama de conexiones 18](#_Toc112630340)

[6.1.1. Sensor de DHT22 18](#_Toc112630341)

[6.1.2. Sensor de humedad del suelo 18](#_Toc112630342)

[6.1.3. Sensor ultrasonido HC-SR04 18](#_Toc112630343)

[6.1.4. Leds 18](#_Toc112630344)

[6.1.5. Caja Portapilas de 6V 19](#_Toc112630345)

[6.1.6. Electrobomba de 5V 19](#_Toc112630346)

[6.1.7. Módulo relé 19](#_Toc112630347)

[6.2. Arduino IDE 20](#_Toc112630348)

[6.3. Node-RED 30](#_Toc112630349)

[6.3.1. Dashboard 31](#_Toc112630350)

[6.3.2. Configuración del servidor MQTT 31](#_Toc112630351)

[6.3.3. Conexión de los nodos 33](#_Toc112630352)

[6.3.4. Leds como avisos 35](#_Toc112630353)

[6.4. Base de datos 37](#_Toc112630354)

[7. Resultados 40](#_Toc112630355)

[7.1. Grupo DHT22 40](#_Toc112630356)

[7.2. Grupo Sensor de humedad del suelo 41](#_Toc112630357)

[7.3. Grupo Control Riego 42](#_Toc112630358)

[7.4. Grupo Sensor ultrasonido HC-SR04 42](#_Toc112630359)

[8. Conclusiones 44](#_Toc112630360)

[9. Bibliografía 45](#_Toc112630361)

[10. Anexos 48](#_Toc112630362)

[10.1. Código en Arduino IDE 48](#_Toc112630363)

[10.2. Estructuración de los datos 54](#_Toc112630364)

[10.3. Imágenes del sistema de riego 55](#_Toc112630365)

# Introducción

En el presente informe buscamos armar un sistema de riego autónomo, que en función de los valores que sean obtenidos por medio de los sensores se pueda iniciar el riego o encenderlo de forma manual si lo queremos, determinar el porcentaje de agua que tengamos en un recipiente o tanque, y avisarnos en caso de que este porcentaje sea demasiado bajo, todo esto se podrá visualizar en la interfaz que sea ha creado con la ayuda de Node-RED que nos mostrara las lecturas de los sensores y el estado de nuestro sistema.

Nuestro proyecto utiliza los siguientes componentes: un módulo relé de un canal, una bomba de agua, NodeMCU con ESP8266, sensor de humedad de agua, DHT22, un sensor ultrasónico HC-SR04, además nos apoyamos sobre dos protoboards para el manejo del circuito y para la creación de dashboard usamos Node-RED, a través del protocolo MQTT podemos obtener la información de nuestros sensores y mostrarlos en la interfaz.

Hay que resaltar que nuestro proyecto se limita a realizar pruebas sobre una maceta, ya que tiene como finalidad integrar todos los conocimientos y realizar un primer acercamiento a lo que sería un sistema de riego para zonas más extensas, logrando así, tener un riego inteligente y autónomo.

# Planteamiento del estudio

## Planteamiento del problema

### Problema general

¿Es posible realizar un sistema de riego autónomo inteligente con los conocimientos adquiridos sobre el Internet de las Cosas?

### Problemas específicos

* ¿Cómo conectar los componentes que se necesitan para aplicar un sistema de riego autónomo inteligente?
* ¿A través de qué plataforma de electrónica se puede programar los componentes para un sistema de riego autónomo inteligente?
* ¿Es posible comunicar hardware para mostrar información de los componentes usados en la aplicación de un sistema de riego autónomo inteligente?

## Objetivos

### Objetivo general

Emplear conocimientos para desarrollar una aplicación en Internet de las Cosas para un sistema de riego autónomo inteligente con cierto nivel de complejidad.

### Objetivos específicos

* Conocer el funcionamiento de los componentes que se usarán en el sistema de riego autónomo.
* Programar componentes que se deben usar para de sistema de riego autónomo inteligente.
* Manejar la herramienta Node-RED para comunicar hardware y mostrar información al probar los componentes usados en la aplicación.

# Marco teórico

## Antecedentes

La agricultura en el Perú es uno de los sectores que más aportan a la economía del País, representa un 6% del PBI nacional, aproximadamente 228 mil millones USD, es por ello que, desde sus inicios, en la agricultura se han implementado diversas técnicas de riego hasta llegar a la actualidad la cual se apoya en la automatización de los procesos.

Chiquito y Paguay (2020) realizaron un proyecto de titulación titulada “Diseño de un prototipo de sistema de riego automatizado mediante una red de sensores que mida la humedad del suelo en los campos agrícolas y permita controlar el consumo del agua” para optar por el título de Ingeniero en Networking y Telecomunicaciones de la Universidad de Guayaquil, proyecto que tuvo como objetivo diseñar e implementar un prototipo de sistema de riego automatizado que sustituya el método tradicional de riego que empleaba la hacienda “Katty Lucy”. La investigación empleo la metodología SCRUM, el proyecto concluye que la implementación de un sistema de irrigación inteligente, bajo parámetros, permitirá regar de forma precisa los cultivos, además de tener un uso adecuado de los recursos.

Camino (2020) realizó un proyecto de titulación, titulada “Análisis y diseño de un prototipo para un sistema de control de riego automatizado con monitoreo y alertas a dispositivos móviles utilizando Arduino, conectividad BLE y SOFTWARE OPEN source para los diferentes cultivos en el Cantón Daule Provincia De Guayas” para optar por el título de Ingeniero en Networking y Telecomunicaciones, proyecto que tuvo como objetivo realizar un prototipo del proceso de riego automatizado el cual incluirá un sistema de monitoreo con BLE y alertas móviles para diferentes cultivos en el cantón Daule, el proyecto utilizó la metodología Cascada. El proyecto concluye que al emplear el prototipo se confirma un considerable ahorro de los recursos de tiempo y de mano de obra, garantizando un riego óptimo.

Cortes y Vargas (2021) realizaron un trabajo de grado, titulado “Diseño e implementación de un sistema de riego automatizado y monitoreo de variables ambientales mediante Iot en los cultivos urbanos de la fundación mujeres empresarias Maria Poussepin” para optar por el título de Ingeniero Electrónico y de Telecomunicaciones de la universidad católica de Colombia, el trabajo tuvo como objetivo desarrollar un sistema de riego automatizado con monitoreo mediante IOT en cultivos urbanos de la Fundación Mujeres empresarias Marie Poussepin. El presente trabajo corresponde a un estudio experimental empleando un diseño metodológico dividido en fases. Se concluyó que al automatizar el riego y tener un control de las variables principales, se optimizó el consumo de agua, además de cumplir con el requisito de monitorear las variables en tiempo real.

# Bases teóricas

## Riego

Se denomina sistema de riego al conjunto de elementos que permiten la aplicación de agua eficiente en una respectiva área de suelo con frecuencia apropiada en cantidades necesarias y en el momento oportuno.

## Métodos de Riego

### Inundación y sumersión

Es un sistema de irrigación que consiste en inundar el terreno con una capa de agua, es el más usado y antiguo, tiene el inconveniente de desperdiciar agua especialmente en campos abiertos debido a la evaporación se dice que la perdida es del 25% aproximadamente, esto sin contar con las filtraciones o con las roturas de los conductos. El agua proviene de un centro de acopio que en muchos casos son pantanos, embalses o tanques especialmente construidos para este propósito, después se distribuye a través de grandes canales, el agua llega por gravedad a las parcelas y terrenos.

### Riego por surcos

Este método consiste en hacer correr el agua por gravedad desde una acequia madre alrededor de pequeñas zanjas o surcos que se forman entre las hileras del cultivo. Los canales son realizados por un trabajador, del tamaño de la apertura de la fuente de

agua y del tamaño de los surcos dependerá la cantidad de agua que se entregue al cultivo. En algunos casos se emplean tuberías al inicio de los surcos que tienen orificios o válvulas por donde sale el agua.

### Riego por goteo

Es un tipo de riego localizado, el agua es aplicada en forma de gotas a través de emisores o de mangueras perforadas, la descarga fluctúa entre 2 a 4 litros por hora. Este método suministra pequeñas cantidades de humedad ala raíz de cada planta, este método garantiza una mínima perdida de agua y evaporación.

Entre sus características más importantes están:

* Permite una considerable reducción de la evaporación del suelo y de las pérdidas por percolación, debido a esto se reduce significativamente las necesidades brutas de agua.
* El agua se infiltra en el terreno, luego se riega en varias direcciones principalmente en dirección vertical y horizontal.
* Solo se moja la parte que se aplica el goteo en contraste solo se humedece el sistema radicular de la planta.

Existen distintos tipos de riego por goteo que se diferencian según su método de aplicación entre los más importantes están:

* Subterráneo: permite el aporte de agua y nutrientes a la planta de forma localizada y bajo la superficie, optimizando el crecimiento de raíces y plantas.
* Superficial: ampliamente utilizado por su amplia flexibilidad.
* Aéreo: se emplea en invernaderos, el agua cae por gravedad al pie de la planta.

## Sistema de automatización de riego

Permite el riego eficiente y óptimo de agua en el suelo. Este compuesto por varios elementos que hacen que las operaciones sean automáticas para comprender un sistema automatizado en un proceso cotidiano se deberá entender conocimientos básicos necesarias que serán empleados en el proyecto.

### Domótica.

Es el uso simultáneo de tecnologías eléctricas, electrónicas aplicadas al control automático de espacios. Pueden estar compuestos por medios de redes externas e internas de comunicación cableadas o inalámbricas.

El sistema domótico está compuesto por tres elementos principales:

* Sensores: captan el cambio físico y transmiten esta información a una unidad de control.
* Actuadores. Trabajan junto a la unidad de control, son de origen hidráulico neumático o eléctrico.
* Unidad de control. Es el principal componente del sistema demótico en él se programan y se reciben órdenes de los demás elementos, está compuesto por interfaces necesarias para para presentar información como pantallas, monitor etc.

### Características del sistema domótico.

Pueden ser centralizados y descentralizados

* Centralizados. Todos los sensores son conectados a la entrada del controlador y los actuadores a la salida, debido a esto toda la información es controlada por la unidad central.
* Descentralizados. Los actuadores y sensores poseen sistema de autocontrol que permite la interacción directa de unos con otros.

## Componentes de un sistema de riego.

### Protoboard

El Protoboard está construido por un bloque central y dos tiras en los extremos. El bloque central está formado por grupos de cinco contactos conectados en común, divididos por una canaleta central, de manera que cuando un componente o dispositivo se inserta en la tablilla, quedan cuatro contactos libres para interconexiones con las terminales del dispositivo.

En las tiras de los extremos hay ocho grupos de 25 contactos comunes, las cuales son convenientes para señales como VCD (voltaje de corriente directa o positiva), GND (tierra o negativo) o cualquier señal que requiera más de cinco contactos comunes. Es recomendable usa terminales o alambre de calibre 24 o 26 para la interconexión, ya que usar alambre de calibre más grueso muy probablemente dañaría los contactos de la tablilla de terminales. Para la interconexión de los elementos del circuito dentro de la tablilla de conexiones, se recomienda preparar alambres descubriendo la parte metálica de los extremos.

### Led

Un diodo LED es un dispositivo que permite el paso de corriente en un solo sentido y que al ser polarizado emite un haz de luz. Trabaja como un diodo normal, pero al recibir corriente eléctrica emite luz. Los LED trabajan aproximadamente con corriente de 2V. Para conectarlos a un voltaje distinto, se debe usar una resistencia.

Su funcionamiento es bastante simple, se conecta la corriente al semiconductor superior del diodo LED lo cual permitirá el paso de corriente eléctrica y hará que el semiconductor emita luz. Según el material del que esté elaborado el semiconductor, los diodos LED proyectarán luces de distintos colores.

### Módulo Relevador de 1 canal

El relé o relevador es un dispositivo electromagnético. Funciona como un interruptor controlado por un circuito eléctrico en el que, por medio de una bobina y un electroimán, se acciona un juego de uno o varios contactos que permiten abrir o cerrar otros circuitos eléctricos independientes.

Especificaciones

* Voltaje entrada: 5 V.
* Voltaje de control:3 ~ 9 V.
* Voltaje de salida: 250 VCA o 30 VDC.
* Corriente a la salida: 10 A.
* Dimensiones: 43 x 17 mm

Terminales

* VCC – Voltaje de entrada de 5 V.
* IN – Voltaje de entrada de 3.3 V.
* GND – Tierra común del circuito.

Bornes de conexión

* N. Abierto
* Común
* N. Cerrado

### Bomba de agua

Esta Bomba de Agua Sumergible, a diferencia de otras, tiene la capacidad de mover líquidos desde un punto a otro, estando dentro del recipiente en donde se desea extraer el contenido. Puede mover líquidos desde un estanque, un recipiente o cualquier fuente de agua hasta cualquier otro punto que se encuentre a una altura no mayor a 40cms de esta bomba. Es posible utilizarla con Microcontroladores tales como Arduino o PIC, e incluso con Microcomputadores como Raspberry, pero es necesario el uso de un Relé o Transistor capaz de maneja su consumo.

Características:

* Voltaje de Operación: 2.5 ~ 6V.
* Corriente de Operación: 130 ~ 220mA.
* Flujo: 80 ~ 120 L/H.
* Elevación máxima de columna de agua: 40 [cm].
* Material: Plástico.
* Diámetro Externo del Tubo de Salida: 7.5 mm.
* Diámetro Interno del Tubo de Salida: 5 mm.
* Largo de cable: 20cm aprox.
* Tamaño: 43mm (largo) x 23mm (diámetro).
* Temperatura del agua: -20 a 50 °C.

### ESP8266

ESP8266 es el nombre de un microcontrolador diseñado por Espressif Systems, una compañía china con sede en Shanghái, este se anuncia a sí mismo como una solución autónoma de redes Wifi que funciona como un puente entre los microcontroladores que hasta ahora existían hasta los MCU con Wifi, siendo además capaz de ejecutar aplicaciones independientes (Ubach Granados, 2017).  
Especificaciones del ESP8266:

* Voltaje de 3.3 V
* Consumo de corriente entre 10 µA y 170 mA.
* Memoria Flash de 16 MB máx. (512 k normal).
* Procesador Tensilica L 106 32bit.
* Velocidad del procesador de 80 a 160 MHz.
* Soporte de 802.11 para b/g/n/d/e/i/k/r.

### Placa de desarrollo NodeMCU

Según Otiniano López (2018) es un dispositivo orientado IoT que posee un chip integrado llamado ESP8266, siendo esta su ventaja principal y que le permite conectarme a internet vía Wifi; es similar a un Arduino en el sentido que integra muchos componentes librerías, código, sensores, etc.

Algunas de las características principales del NodeMCU son:

* Conversor Serie-USB para poder programar y alimentar a través del USB.
* Fácil acceso a los pines.
* Pines de alimentación para sensores y componentes.
* LEDs para indicar estado.
* Botón de Reset y Flash.

### Sensor de humedad de suelo

Según Cabrera y Montes (2021), este sensor se encarga de medir la cantidad de humedad presente en el suelo, para esto utiliza dos electrodos que pasan corriente a través del suelo, leyendo así la resistencia, ya que al presentarse cantidades enormes de agua permiten que la tierra conduzca la electricidad fácilmente (menos resistencia), pero si el suelo se encuentra seco, no es posible que se conduzca la electricidad fácilmente (mayor resistencia). Especificaciones del sensor de humedad de suelo:

* Voltaje de entrada entre 3.3 a 5 V.
* Voltaje de salida de 0 a 4.2 V.
* Corriente de 35 mA
* VCC (Tensión de alimentación).
* GND (Tierra).
* A0: Salida analógica que entrega una tensión proporcional a la humedad.
* D0: Salida digital.

### Sensor DHT22

Es un sensor digital que como su nombre lo indica se encarga de medir la temperatura y la humedad del ambiente, siendo su rango de medición de -40°C a 80 °C con precisión de ±0.5 °C, donde este trabaja a través de un protocolo serial donde se envían los datos por medio del pin de datos (Laverde Mena y Laverde Mena, 2021).  
Especificaciones del sensor DHT22:

* Alimentación de 5V.
* Rango de temperatura entre -40°C a 80 °C.
* Margen de error de temperatura ± 0.5.
* Rango de humedad entre 0 - 100% RH.
* Margen de error de humedad ± 2.

### Sensor de ultrasonido HC-SR04

El sensor HC-SR04 es un sensor de distancia de bajo costo que utiliza ultrasonido para determinar la distancia de un objeto en un rango de 2 a 450 cm. Destaca por su pequeño tamaño, bajo consumo energético, buena precisión y excelente precio. El sensor HC-SR04 es el más utilizado dentro de los sensores de tipo ultrasonido, principalmente por la cantidad de información y proyectos disponibles en la web. De igual forma es el más empleado en proyectos de robótica como robots laberinto o sumo, y en proyectos de automatización como sistemas de medición de nivel o distancia.

El sensor HC-SR04 posee dos transductores: un emisor y un receptor piezoeléctricos, además de la electrónica necesaria para su operación. El funcionamiento del sensor es el siguiente: el emisor piezoeléctrico emite 8 pulsos de ultrasonido(40KHz) luego de recibir la orden en el pin TRIG, las ondas de sonido viajan en el aire y rebotan al encontrar un objeto, el sonido de rebote es detectado por el receptor piezoeléctrico, luego el pin ECHO cambia a Alto (5V) por un tiempo igual al que demoró la onda desde que fue emitida hasta que fue detectada, el tiempo del pulso ECO es medido por el microcontrolador y asi se puede calcular la distancia al objeto. El funcionamiento del sensor no se ve afectado por la luz solar o material de color negro (aunque los materiales blandos acústicamente como tela o lana pueden llegar a ser difíciles de detectar).

Especificaciones Técnicas:

* Voltaje de Operación: 5V DC.
* Corriente de reposo: < 2mA.
* Corriente de trabajo: 15mA.
* Rango de medición: 2cm a 450cm.
* Precisión: +- 3mm.
* Ángulo de apertura: 15°.
* Frecuencia de ultrasonido: 40KHz.
* Duración mínima del pulso de disparo TRIG (nivel TTL): 10 μS
* Duración del pulso ECO de salida (nivel TTL): 100-25000 μS
* Dimensiones: 45\*20\*15 mm
* Tiempo mínimo de espera entre una medida y el inicio de otra 20ms (recomendable 50ms).

## Protocolo de comunicación MQTT

MQTT significa “Message Queue Telemetry Transport”, este se basa en utilizar la mensajería publicación/suscripción entre máquinas en el concepto de IoT. Este protocolo se enfoca principalmente en la comunicación entre sensores, por lo que está pensado para ser simple, con fácil implementación, ligero y utilizar muy poco ancho de banda. De esta manera el coste computacional entre la comunicación entre máquinas es bajo y por lo tanto se necesitan pocos recursos (Calleja Collado y Guasch Llobera, 2019).

### Arquitectura

MQTT presenta una topología en forma de estrella donde el nodo principal tiene la capacidad de gestionar un gran número de clientes, este nodo principal es conocido como “Broker” y los elementos conectados a este elemento se les conoce como clientes debido a que dependen en su totalidad al nodo principal (Calleja Collado y Guasch Llobera, 2019).

### Broker

El “Broker” se encarga de distribuir los mensajes entre los receptores que se hayan suscrito al “topic” (tema del mensaje), de esta manera los receptores (clientes) solo reciben los mensajes que van dirigidos a los “topics” que están suscritos (Calleja Collado y Guasch Llobera, 2019).

### Cliente

Un cliente puede ser tanto publicador como suscriptor, o ambos; y tienen como principal característica de ser independientes unos de otros, por tanto, no necesitan saber si se encuentra alguien en el otro lado, simplemente transmiten la información. Esta característica permite crear proyectos con una mayor escalabilidad (Calleja Collado y Guasch Llobera, 2019).

## Node – RED

Según Benedicto (2022), es una herramienta de código abierto utilizada en IoT que sirve para comunicar hardware y los servicios que permiten dibujar el comportamiento de una aplicación como una red de nodos, en donde cada uno de estos recibe datos que son tratados y posteriormente enlazados, permitiendo el manejo de flujos de transformación entre mensajes relacionados a tópicos de MQTT y mostrándolas a través de una interfaz amigable y entorno de trabajo predefinido lo que permite la sencillez en su

# Componentes del sistema de riego

Ahora pasando a explicar los componentes de nuestro sistema de riego empezaremos mencionando los sensores que se están utilizando, estos son el sensor DHT22, el sensor de humedad del suelo y sensor de ultrasonido HC-SR04.

El sensor DHT22 es usado para obtener los valores de la temperatura (en Celsius) y la humedad; el sensor de humedad del suelo, como su nombre lo dice, se usará para medir la humedad del suelo presente en nuestro ambiente de trabajo (maceta, suelo, etc.); y el sensor ultrasonido HC-SR04 es usado para conocer el porcentaje de agua que tenemos en nuestro almacén de agua que luego es usada para regar a las plantas.

El cálculo del porcentaje de agua se logra al considerar las propiedades que el sensor ultrasonido posee, es decir, gracias a que este nos permite conocer el tiempo que le toma al pulso de sonido llegar a un obstáculo, entonces se puede calcular la distancia de forma sencilla recordando que la velocidad del sonido recorre aproximadamente 1 centímetros en 29.15 microsegundos, por lo que, si yo sé cuál es la distancia que hay cuando el almacén está vacío, y la distancia cuando está lleno entonces dependiendo de la distancia que calculamos gracias el tiempo que arroja el sensor es posible obtener una aproximación de que porcentaje representa la cantidad de agua que tengamos actualmente en nuestro almacén de agua.

Otro componente importante son los 2 leds con los que contamos, el led 1 está altamente relacionado con la activación de la bomba de agua y el riego autónomo, ya que este led funciona como una señal para el usuario de que se activara el riego, el cual se encenderá 5 segundos antes de que el riego inicie, por lo que sí es aplicado a un área más grande el tener un aviso de que se iniciara el riego permitiría evitar ponerse en camino del riego y terminar mojándose; el led 2 está relacionado con el porcentaje de agua de nuestro almacén, este se activara si el porcentaje de agua es menor al 20%, entonces el usuario al visualizar que el led se encienda sabe cuál es un momento adecuado para reincorporar más agua en el almacén. El estado de los leds también se puede visualizar en el dashboard de Node-RED, por si el usuario quiere visualizarlo desde ahí.

Por último, tenemos la caja portapilas de 6V (para 4 pilas de 1.5V), el módulo relé y la electrobomba de 5V, estos serán conectados entre sí y permitirán que se el módulo ESP8266 pueda controlar en encendido de la electrobomba y así poder realizar el riego autónomo de manera efectiva.

Las conexiones entre los componentes electrónicos del sistema se presentarán detalladamente en el capítulo de implementación del sistema.

# Implementación del sistema

Para la implementación del sistema se utilizó tanto Arduino IDE como Node-RED, a su vez se realizaron las conexiones respectivas de todos los componentes electrónicos.

## Diagrama de conexiones

Primero se definieron los pines que se utilizaría para las conexiones:

### Sensor de DHT22

* Los pines VCC y GND del sensor fueron conectados a los pines 3V3 y GND respectivamente del módulo ESP8266.
* El pin Out se conectó al pin digital D2 (GPIO 4).

### Sensor de humedad del suelo

* Los pines VCC y GND del sensor fueron conectados a los pines 3V3 y GND respectivamente.
* El pin A0 se conectó al único pin analógico (A0) del ESP8266.

### Sensor ultrasonido HC-SR04

* Los pines VCC y GND del sensor fueron conectados a los pines 3V3 y GND respectivamente.
* El pin Trig se conectó al pin digital D7 (GPIO 13).
* El pin Echo se conectó al pin digital D8 (GPIO 15).

### Leds

* Los pines negativos (cátodos) de ambos leds se conectaron a un pin GND.
* Los pines positivos (ánodos) del led 1 y led 2se conectaron a los pines digitales D5 (GPIO 14) y D6 (GPIO 12) respectivamente, además se usaron dos resistencias de 220 Ohm.

### Caja Portapilas de 6V

* El cable positivo (rojo) se conectó con el cable positivo de la electrobomba.
* El cable negativo (negro) se conectó con el borne de conexión NO (normalmente abierto) del módulo relé.

### Electrobomba de 5V

* El cable positivo (rojo) se conectó con el cable positivo de la caja portapilas.
* El cable negativo (negro) se conectó con el borne de conexión COM (Común) del módulo relé.

### Módulo relé

* Los bornes de conexión NO y COM se conectan con los cables negativos de la caja portapilas y la electrobomba respectivamente.
* Los pines GND y VCC se conectaron a los pines GND y 3V3 del módulo ESP8266.
* El pin IN se conecta con el pin digital D1 (GPIO 5).

El diagrama de conexiones general se presenta a continuación:

Diagrama, Esquemático

Descripción generada automáticamente

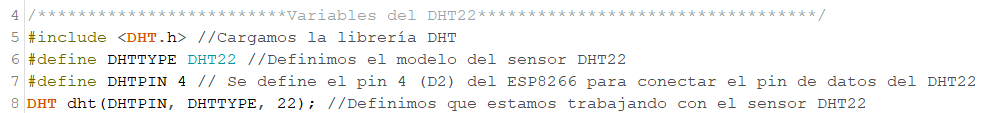
## Arduino IDE

Se utilizo Arduino IDE para el desarrollo del código, en el cual se realizarán las configuraciones respectivas de los sensores y se programaron las diversas funcionalidades de nuestra aplicación; este código será explicado paso por paso a continuación.

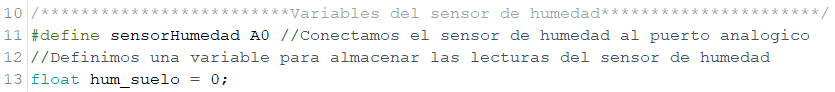
Lo primero es instalar las librerías ESP8266WiFi y la librería PubSubClient, la primera agrega las funciones necesarias para conectarnos a nuestra red wifi, y la segunda para trabajar con el protocolo MQTT.



Luego procedemos a definir las variables necesarias para trabajar con el sensor DHT22, se agregó la librería DHT y se definió el tipo de sensor, en este caso el DHT22, además de definir con que pin trabajaremos (D2).



Ahora para el sensor de humedad del suelo definimos que trabajaremos el único pin analógico del módulo ESP8266 (A0), y se crea una variable “hum\_suelo” donde se almacenará la humedad del suelo.



Se creo una variable entera donde se define en que pin conectaremos (D1) el módulo relé que permitirá encender la bomba e iniciar el riego.



Para usar el sensor ultrasonido se define primero variables que indican a que pines del modulo ESP8266 se conectaran, en este caso el pin Trig se conecta al D7 y el Echo al pin D8. Además, como se ve en la imagen, en la línea 25 y 26 se crearon dos variables de tipo float para almacenar el tiempo y la distancia. También haremos uso de un arreglo de 100 elementos, en donde almacenaremos las lecturas (valores de distancia) que nos otorga el sensor ultrasonido, y así luego calcular el promedio de estas distancias. Y por último, se crea una variable “distanciaLleno” donde se ingresa la distancia que hay desde el sensor ultrasonido al nivel del agua del recipiente para que se considere llena, otra variable “distanciaVacio” para la distancia desde el sensor al fondo del recipiente de agua, estos valores son definidos por nosotros y dependen del recipiente donde estemos almacenando el agua; y por último una variable a la que se le asignara el porcentaje de agua que luego se mostrara en el dashboard de Node-RED

Interfaz de usuario gráfica, Texto

Descripción generada automáticamente

Se crean dos variables donde se establece los pines a los que conectaremos nuestros dos leds, D5 para el led 1 y D6 para el led 2.



Se definen los parámetros para la conexión WIFI, como el nombre de nuestra red y la contraseña.



Se define los parámetros para la conexión MQTT, como el servidor, el usuario y la contraseña. Luego se crean dos String, uno donde se almacenará el topic y otro donde se almacenará el payload, además se crean 4 arreglos tipo char en donde posteriormente se almacenarán los valores que serán publicados en los topics respectivos;

Texto

Descripción generada automáticamente

Creamos una variable llamada lastMsg que será usado para que los datos sean publicados en los topic cada 5 segundos, además en la línea 67 tenemos un String que será usado para conocer el estado de la bomba.

Imagen que contiene Texto

Descripción generada automáticamente

Creamos una función a la que llamaremos setup\_wifi, esta nos permitirá conectarnos a nuestra red wifi, para esto son necesarios las credenciales de nuestra red que fueron definidos anteriormente, es decir, el nombre de nuestra red y su contraseña.

Interfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamente

Se crea una función callback, en la que almacenamos nuestro topic en la variable “mensaje” y también almacenamos el payload en la variable “inform”.

Texto

Descripción generada automáticamente

Creamos una función reconnect que será usada cada vez que nos desconectemos y así reintentar la conexión MQTT y luego nos subscribimos a los topics correspondientes, en caso de que la conexión sea fallida se imprimirá un mensaje que dice que se intentara de nuevo dentro de 5 segundos.

Interfaz de usuario gráfica, Texto, Aplicación, Correo electrónico

Descripción generada automáticamente

En la función setup hacemos las configuraciones respectivas para cada uno de los dispositivos que conforman nuestra aplicación. Primero el pin donde se conecta el módulo relé y que hemos llamada bombaAgua se establece como salida, luego para el sensor ultrasonido se establece el pinTring como salida y el pinEcho como entrada, esto es porque el pinTrig será usado emitir un pulso de sonido y el pinEcho para obtener el tiempo; los leds por otro lado ambos son de salida. Para configurar el sensor DHT22 solo es necesario usar el método begin(), y para el sensor de humedad del suelo se define como entrada ya que a partir del valor que obtendremos de este nos es posible conocer el porcentaje de humedad del suelo. Por último, iniciamos la comunicación serial, llamamos a la función setup\_wifi.

Texto

Descripción generada automáticamente

En la función loop lo primero que hacemos es obtener los valores de los sensores DHT22, para eso dentro de las variables “tem” y “hum” almacenamos la temperatura y la humedad con ayuda de las funciones readTemperature() y readHumedity() como podemos ver en las líneas 168 y 170. Para el sensor de humedad del suelo lo que se hace es obtener el valor de la lectura analógica y a partir de esto con algunas operaciones matemáticas obtenemos el porcentaje de humedad del suelo.

Texto

Descripción generada automáticamente

Luego para el sensor ultrasonido configuramos el trigger para generar un pulso limpio, para eso primero se pone el pinTrig como LOW por 4 microsegundos, después como HIGH por 10 microsegundos para posteriormente ponerlo de nuevo en LOW. Una vez se realizó eso se calculó el tiempo que le toma al pulso de sonido llegar al obstáculo, para esto se hace uso de la línea de condigo 186.

Imagen que contiene Texto

Descripción generada automáticamente

Se calcula la distancia, esto se hace gracias al tiempo que hemos obtenido y a que sabemos que la velocidad del sonido es de 1cm en 29.15us, una vez que obtenemos la distancia esta se almacena en el arreglo de lecturas, una vez nuestro arreglo ya tenga 100 lecturas de distancia se calcula el promedio, como se ve en la línea 202, para esto se hace uso de la función promedio() que se creó. Por último, se calcula el porcentaje de agua que tenemos en el recipiente, esto se logra gracias a que como conocemos la distancia que hay desde el sensor hasta el nivel de agua del recipiente y las distancia cuando está lleno y cuando esta vacio, y con unas simples operaciones matemáticas, como se ve en la línea 205, es posible calcular el porcentaje de agua, y si resulta que este porcentaje de agua es menor al 20% entonces se encenderá el led 2, sino este estará apagado.

Texto

Descripción generada automáticamente

Si es que se produce una desconexión entonces llamaremos a la función reconnect para volver a establecer la conexión MQTT, luego de eso llamamos a la función riego\_automatico y se le envían como parámetros la temperatura, la humedad y la humedad del suelo.

Interfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamente

Aquí lo que hacemos es publicar los valores de la temperatura, la humedad, la humedad del suelo y el porcentaje de agua en los topics respectivos, además estos valores se imprimen para que sean visualizados en el monitor serial del Arduino IDE.

Texto, Aplicación

Descripción generada automáticamente

Creamos una función llamada riego\_automatico que recibirá como parámetros la temperatura, la humedad y la humedad del suelo, y si resulta que se cumple la condición que se visualiza en la línea 271 entonces se procederá a encender el led 1; 5 segundos después se encenderá la bomba de agua y se iniciará el riego. De igual manera, si es que en el dashboard de Node-RED el usuario presiona el botón de encendido de riego entonces la bomba también se encendería, caso contrario siempre se mantendrá apagada.

Interfaz de usuario gráfica, Texto, Aplicación, Correo electrónico

Descripción generada automáticamente

Por último, tenemos nuestra función para calcular el promedio y que se usó en el loop para calcular el promedio de las distancias.

Interfaz de usuario gráfica, Texto, Aplicación, Correo electrónico

Descripción generada automáticamente

## Node-RED

Para el control y monitoreo de nuestro sistema de riego se utilizó la plataforma de Node-RED, dentro de esta trabajamos con diversos nodos y realizamos diversas conexiones que se explicaran a continuación de manera detallada.

### Dashboard

Para poder trabajar con dashboard en Node-RED es necesario instalar esta funcionalidad dentro del sistema, una vez hecho esto se procedió a la creación de un tab, al cual se le asigno el nombre de “PROYECTO IOT”, dentro de este se crearon 4 group que será necesarios para presentar una interfaz más ordenada y asignar los nodos del dashboard dentro de esta.

Interfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamente

### Configuración del servidor MQTT

Para que podamos recibir las lecturas de los sensores es necesario configurar la conexión MQTT dentro de Node-RED, para esto dentro de los nodos mqtt es necesario agregar un nuesvo mqtt-broker, para esto primero en la pestaña conexión se ingresa el servidor (ip) de nuestra PC y con la estableceremos la conexión, luego el puerto, en nuestro caso no es necesario que cambiemos el puerto, y por último el Client ID.

Interfaz de usuario gráfica, Aplicación

Descripción generada automáticamente

Además, en la pestaña de seguridad es posible establecer un username y una contraseña si es que así se desea. Una vez realizado esto se da clic en el botón Add para finalizar con la configuración.

Interfaz de usuario gráfica, Texto, Aplicación, Correo electrónico

Descripción generada automáticamente

Finalmente, nuestra configuración quedaría de la siguiente manera, donde nuestro username es “admin” y nuestra contraseña “12345”.

Interfaz de usuario gráfica, Aplicación

Descripción generada automáticamente

### Conexión de los nodos

Ahora se presentarán todos los nodos, sus conexiones y las funciones de estos:

#### Encendido Manual del riego:

Se agrego un nodo button al que llamaremos “Encender riego”, este será usado para controlar el encendido manual de la bomba con solo presionarlo, El nodo button estará conectado con un nodo mqtt-out con el topic “control\_riego” y que ira publicando el valor 1 cada vez que se presiona el botón, y esto hará que la bomba se encienda.

Imagen que contiene Gráfico

Descripción generada automáticamente

A su vez, con ayuda del nodo function “almacenar\_estadoBomba” podremos guardar esta variable como una variable tipo Flow que se podrá recuperar para ser usada posteriormente.

Interfaz de usuario gráfica, Texto, Aplicación, Correo electrónico

Descripción generada automáticamente

#### Lecturas del sensor DHT22

Para poder mostrar los valores obtenidos por el sensor DHT22 en el dashboard de Node-RED utilizaremos dos nodos mqtt-in con los topics “dht22-tem” y “dht22-hum” para la temperatura y la humedad respectivamente; estos nodos irán conectados a un nodo text, un nodo gauge y un nodo chart, lo que permitirán una mejor visualización de los datos en el dashboard, y de forma similar que con el botón de encendido del riego también se usaran funciones para almacenar los valores de la temperatura y la humedad dentro de variables de tipo Flow.

Diagrama

Descripción generada automáticamente

#### Lecturas del sensor de humedad del suelo

De forma similar que con las lecturas del sensor DHT22 se usara un nodo mqtt-in con el topic “sensor-hum\_suelo” para recibir los valores de la humedad del suelo, también se usó un nodo text, un nodo gauge y un nodo chart para visualizar la información de manera más ordenada. También tendremos una función que almacenara la humedad del suelo dentro de un variable de tipo Flow.

Diagrama

Descripción generada automáticamente

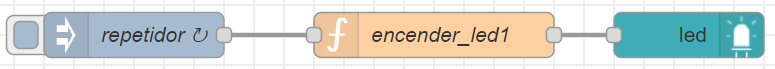
#### Lecturas del porcentaje de agua

Como se mencionó anteriormente el sensor ultrasonido será usado para conocer el porcentaje de agua que presenta nuestro almacén de agua, en nuestro caso un valde con agua, y estos valores serán publicados en el topic sensor-ultrasonido, por eso se agregó un nodo mqtt-in con ese topic y otros nodos (text, gauge y chart) para que se puedan ir mostrando los valores del porcentaje de agua de la misma manera que se visualizan los valores arrojados por los sensores. Además, se usó un nodo function que se explicara en un momento.

### Leds como avisos

Dentro de nuestro dashboard, poseemos dos nodos led, estos son usados como una especie de aviso o alarma para el usuario.

Primero tenemos el nodo LED 1, que se encuentra dentro del grupo de Control Riego junto con el botón de encendido manual, este avisara al usuario cuando se esté dando el riego en nuestro sistema, para esto hace uso de un nodo function, llamado encender\_led1, que hace uso de las variables de tipo Flow, que se pueden visualizar en el Context Data, y que si se cumple la condición se retornara 1 que hará que se encienda el led.



Interfaz de usuario gráfica, Texto, Aplicación, Correo electrónico

Descripción generada automáticamente

Para el nodo LED 2 no es necesario almacenar el valor del porcentaje de agua dentro de una variable de tipo Flow, solo es necesario conectarlo al nodo mqtt-in con el topic “sensor-ultrasonido”, por lo que con el nodo function “encender\_led2” hacemos que si el porcentaje de agua es menor al 20% entonces retornara 1 que hará que se encienda en el dashboard este led.

Diagrama

Descripción generada automáticamente

Interfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamente

## Base de datos

Para almacenar los datos usaremos el programa CoolTermWin que se puede descargar en el siguiente link: <https://freeware.the-meiers.org/>

Dentro de este programa lo que debemos hacer es clic en Options y configurar el puerto y el baudrate, además podemos darle clic en Receive y marcar la opción de “Add Timestamps to received data” para almacenar la fecha y hora de cuando recibimos los datos.

Interfaz de usuario gráfica, Texto, Aplicación, Correo electrónico

Descripción generada automáticamente

Interfaz de usuario gráfica, Aplicación

Descripción generada automáticamente Interfaz de usuario gráfica, Aplicación

Descripción generada automáticamente

Y si queremos guardar los datos en un .txt debemos dar clic en la pestaña Connection y luego en Capture to Text/Binary File.

Interfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamente

Una vez tengamos un txt con los datos ya podemos pasarlos estos a un csv y almacenar todas las lecturas que tenemos.

# Resultados

Se presentará el dashboard desarrollado en la plataforma de Node-RED con todos los gráficos que se decidieron agregar divididos en 4 grupos.

Interfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamente

## Grupo DHT22

Aquí se podrán visualizar los valores de la temperatura y humedad que serán obtenidos por el sensor DHT22, estas se podrán visualizar en un gauge y en un gráfico que permite ver cómo han variado a lo largo del tiempo. Los valores de la temperatura pueden ir entre -40 y 80, y la humedad, al ser relativa, va desde 0 a 100.

Tabla

Descripción generada automáticamente

## Grupo Sensor de humedad del suelo

En este grupo se visualizarán las lecturas del sensor de humedad del suelo, donde estos valores pueden ser desde 0 a 100, y se usará un gauge y un gráfico para que la visualización de los datos sea más dinámica.

Interfaz de usuario gráfica, Aplicación

Descripción generada automáticamente

## Grupo Control Riego

Aquí se visualiza el botón de encendido manual del riego, así como el LED 1 que sirve como aviso de si el riego se encuentra activo o no, en caso el riego no se encuentre activo se visualizara un color gris en el LED 1, y si se encontrase activo presentara un color verde.

Interfaz de usuario gráfica, Aplicación

Descripción generada automáticamente

## Grupo Sensor ultrasonido HC-SR04

Se presentan los valores del porcentaje de agua de nuestro almacén en los mismos gráficos que usamos para representar las lecturas de los sensores, además se posee un LED 2 a modo de aviso para el usuario, si el porcentaje de agua es mayor igual al 20% entonces este tendrá un color verde, en caso contraria tendrá un color rojo a modo de alarma de que el nivel de agua es muy bajo.

Diagrama

Descripción generada automáticamente

Estos valores concuerdan con los que podemos observar al visualizar nuestro monitor serie en Arduino IDE, por ejemplo, como se ve en la imagen siguiente:

Interfaz de usuario gráfica, Texto

Descripción generada automáticamente

# Conclusiones

1. El grupo llego a la conclusión que el sistema de riego funciona de forma autónoma en base a las lecturas que se obtienen por medio de los sensores, además es posible monitorear nuestro sistema gracias al dashboard que fue desarrollado en Node-RED.
2. Se concluyo que el sistema de riego autónomo puede ser implementado en un ambiente real, como un parque, un jardín, etc., con los cuidados correspondientes, para que el cableado y los componentes electrónicos, además se evidencio que al estar trabajando con agua es necesario contar con medios de almacenamiento y transporte de agua más seguros y con materiales más resistentes, además de una mejora en el tipo de bomba que se está utilizando si es que se trabaja en un área más extensa.
3. Concluimos que el sistema de riego permite que se le agreguen componentes adicionales para agregar funcionalidades extras al sistema, esto gracias a las libertades que nos permite trabajar con el módulo ESP8266, esto se evidencio con la incorporación del sensor ultrasonido que se realizó en nuestro sistema.

# Bibliografía

Benedicto Basallote, D. (2022). *Diseño de una maleta portátil para realizar demostraciones de procesos de digitalización mediante Node-RED contextuados en la industria 4.0*. [Trabajo final de Máster, Universidad Politécnica de Cataluña]. Repositorio de la Universidad Politécnica de Cataluña. <https://upcommons.upc.edu/handle/2117/367568>

Calleja Collado, M. y Guash Llobera, J. (2019). *Monitorización de sensores con arduino utilizando el protocolo MQTT.* [Trabajo final de grado, Universidad Politécnica de Cataluña]. Repositorio de la Universidad Politécnica de Cataluña. <https://upcommons.upc.edu/handle/2117/134193>

Cabrera Caisa, S. G., y Montes Estrada, I. O. (2021). *Implementación de un sistema automatizado para riego basado en la tecnología arduino para controlar balance de humedad de suelo en el recinto Siete Ríos*. [Proyecto de investigación, Universidad Técnica de Cotopaxi]. Repositorio de la Universidad Técnica de Cotopaxi. <http://repositorio.utc.edu.ec/handle/27000/7302>

Chiquito, R., & Paguay, C. (2020). Diseño de un prototipo de sistema de riego automatizado mediante una red de sensores que mida la humedad del suelo en los campos agrícolas y permita controlar el consumo de agua. Guayaquil.  
<http://repositorio.ug.edu.ec/handle/redug/48816>

Camino, J. (2020). Análisis y diseño de un prototipo para un sistema de control de riego automatizado con monitoreo y alertas a dispositivos móviles utilizando arduino, conectividad BLE y SOFTWARE OPEN source para los diferentes cultivos en el Cantón Daule Provincia De Guay. Guayaquil: Universidad de Guayaquil. Facultad de Ciencias Matemáticas y Físicas. Carrera de Ingeniera en Networking y Telecomunicaciones. Obtenido de <http://repositorio.ug.edu.ec/handle/redug/48784>

Cortes, V., & Vargas, M. (2021). Diseño e implementación de un sistema de riego automatizado y monitoreo de variables ambientales mediante Iot en los cultivos urbanos de la fundación mujeres empresarias Maria Poussepin. Bogotá: Universidad Católica de Colombia. Obtenido de <https://hdl.handle.net/10983/25546>

Laverde Mena, J. A., & Laverde Mena, C. G. (2021). Internet de las cosas aplicado en la agricultura ecuatoriana: Una propuesta para sistemas de riego. *Revista Dilemas Contemporáneos: Educación, Política y Valores*. 2(31). <https://doi.org/10.46377/dilemas.v8i2.2542>

Otiniano López, M. F. (2018). *Sistema de Medición Acústica usando NODEMCU ESP8266 para Determinar el Nivel de Ruido en Av. Víctor Larco cuadra 14 Trujillo 2018*. [Tesis para para el título de Ingeniero De Sistemas, Universidad César Vallejo]. Repositorio de la Universidad César Vallejo. <https://repositorio.ucv.edu.pe/handle/20.500.12692/28240>

Ubach Granados, C. (2017). Programando directamente un ESP8266. [Tesis de Bachiller, Universidad Politécnica de Cataluña]. Repositorio de la Universidad Politécnica de Cataluña. <http://hdl.handle.net/2117/105042>

# Anexos

## Código en Arduino IDE

#include <ESP8266WiFi.h> //libreria que proporciona las rutinas especificas Wifi de ESP8266

#include <PubSubClient.h> //Libreria para enviar y recibir mensajes MQTT

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*Variables del DHT22\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

#include <DHT.h> //Cargamos la librería DHT

#define DHTTYPE DHT22 //Definimos el modelo del sensor DHT22

#define DHTPIN 4 // Se define el pin 4 (D2) del ESP8266 para conectar el pin de datos del DHT22

DHT dht**(**DHTPIN**,** DHTTYPE**,** 22**);** //Definimos que estamos trabajando con el sensor DHT22

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*Variables del sensor de humedad\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

#define sensorHumedad A0 //Conectamos el sensor de humedad al puerto analogico

//Definimos una variable para almacenar las lecturas del sensor de humedad

float hum\_suelo **=** 0**;**

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*Para la bomba de agua\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

int bombaAgua **=** 5**;** //Indica el pin digital donde se conecta rele (D1)

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*Para el sensor ultrasonido HC-SR04\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

//Configuramos los pines del sensor Trigger y Echo

int PinTrig **=** 13**;** //Pin (D7) donde conectaremos el pin trigger del sensor ultrasonido

int PinEcho **=** 15**;** //Pin (D8) donde conectaremos el pin echo del sensor ultrasonido

//Creamos una variable tiempo donde almacenaremos el tiempo que le toma a la onda de sonido

//recorrer una distancia a la velocidad del sonido

float tiempo**;**

float distancia**;** //Se calcula con ayuda del tiempo

#define nLecturas 100 //Tamaño del arreglo de lecturas que crearemos

float lecturas**[**nLecturas**];** //Arreglo donde almacenaremos las lecturas

int nActual **=** 0**;** //Indice de la lectura en la que nos encontramos

//Creamos dos variables

float distanciaLleno **=** 3.5**;** //Distancia que indica que el recipiente de agua esta lleno

float distanciaVacio **=** 17.5**;** //Distancia que indica que el recipiente de agua esta vacio

//Creamos una variable donde se asignara el porcentaje de agua

//de nuestro recipiente (el almacen de agua)

float porcentajeAgua**;**

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*Para los 2 LED que usaremos\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

int LED1 **=** 14**;** //Se encendera cuando la bomba de agua se active y estara encendida hasta que esta se apague

int LED2 **=** 12**;** //Estara encendido cuando el porcentaje de agua de nuestro recipiente es menor al 20%

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*Parametros de conexion Wifi\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

//Depende de cada maquina

const char**\*** ssid **=** "Sergio"**;** //Nombre de la red con la que trabajamos

const char**\*** password **=** "07503108"**;** //Contraseña del la red con la que trabajamos

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*MQTT\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

const char**\*** mqtt\_server **=** "192.168.0.5"**;** //Dirección ip del dispositivo

const char**\*** mqtt\_user **=** "admin"**;** //Ingresamos un usuario

const char**\*** mqtt\_pass **=** "12345"**;** //Ingresamos una contraseña

String mensaje**;** //Para almacenar el topic

String inform**;** //Para almacenar el payload

//Creamos unas variables para almacenar los mensajes que seran posteriormente publicados

char dht22tem**[**15**],** dht22hum**[**15**],** sensorhum**[**15**],** sensorultrasonic**[**15**];**

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

// Crea una clase WiFiClient para conectarse al servidor MQTT

WiFiClient espClient**;**

PubSubClient client**(**espClient**);**

long lastMsg **=** 0**;** //Definimos una variable donde se almacenara el momento en donde se envio el ultimo mensaje

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

//Asiganamos una variable para almacenar el estado de la bomba

String estadoBomba **=** "0"**;**

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

void setup\_wifi**()** **{**

delay**(**10**);**

// Empezamos conectándonos a una red WiFi

Serial**.**println**();**

Serial**.**print**(**"Conectado a "**);**

Serial**.**println**(**ssid**);**

//Establecemos el inicio de la conexión con los credenciales del Wifi

WiFi**.**begin**(**ssid**,** password**);**

**while** **(**WiFi**.**status**()** **!=** WL\_CONNECTED**)** **{** //Mientras no este conectado imprimimos un punto

delay**(**500**);** //Esperamos 500 milisegundos

Serial**.**print**(**"."**);**

**}**

Serial**.**println**(**""**);**

//Despues de lograrse la conexion imprimos un mensaje

Serial**.**println**(**"Conectado a la red WiFi con dirección IP: "**);**

Serial**.**println**(**WiFi**.**localIP**());** //Imprimimos la dirección IP de la red Wifi

**}**

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

//Funcion para comprobar si recibe el topic

void callback**(**char**\*** topic**,** byte**\*** payload**,** unsigned int length**)** **{**

mensaje **=** topic**;** //Almacenamos el topic en la variable mensaje

inform **=** ""**;** //String donde se almacenara el payload

//Comentaremos los siguientes mensajes

//Serial.print("El mensaje llegó [");

//Serial.print(mensaje); //imprimimos el mensaje

//Serial.print("] ");

**for** **(**int i **=** 0**;** i **<** length**;** i**++)** **{**

inform **+=** **(**char**)**payload**[**i**];** //Guardamos el payload en un string

**}**

//Serial.print(inform); //Imprimos el string que contiene al payload

//Serial.println();

**}**

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

void reconnect**()** **{** //Funcion para reconectarnos

// Loop hasta que estemos reconectados

**while** **(!**client**.**connected**())** **{**

Serial**.**print**(**"Intentando conexion MQTT..."**);**

**if** **(**client**.**connect**(**"ESP8266Client"**,**mqtt\_user**,** mqtt\_pass**))** **{** //Si estamos conectados, entonces...

//Publicamos en cada uno de los topic el mensaje Conectado

Serial**.**println**(**"Conectado"**);**

client**.**publish**(**"dht22-tem"**,** "Conectado"**);**

client**.**publish**(**"dht22-hum"**,** "Conectado"**);**

client**.**publish**(**"sensor-hum\_suelo"**,** "Conectado"**);**

client**.**publish**(**"sensor-ultrasonido"**,** "Conectado"**);**

client**.**publish**(**"control\_riego"**,** "Conectado"**);**

//Nos suscribimos a los topicos

client**.**subscribe**(**"dht22-tem"**);**

client**.**subscribe**(**"dht22-hum"**);**

client**.**subscribe**(**"sensor-hum\_suelo"**);**

client**.**subscribe**(**"sensor-ultrasonido"**);**

client**.**subscribe**(**"control\_riego"**);**

**}** **else** **{** //si falla la conexión se imprime un mensaje

Serial**.**print**(**"Conexion fallida, rc="**);**

Serial**.**print**(**client**.**state**());** //Se imprime la condicion del cliente

Serial**.**println**(**"....pruebe de nuevo dentro de 5 segundos...."**);**

delay**(**5000**);** //Esperamos 5 segundos

**}**

**}**

**}**

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

void setup**()** **{**

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*Para la bomba de agua\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

pinMode**(**bombaAgua**,** OUTPUT**);** //Establecemos el pin 8 como salida

/\*\*\*\*\*\*\*\*\*\*\*\*Para el sensor ultrasonido HC-SR04\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

// Ponemos el pin Trig en modo salida

pinMode**(**PinTrig**,** OUTPUT**);** //porque de aqui saldra un pulso

// Ponemos el pin Echo en modo entrada

pinMode**(**PinEcho**,** INPUT**);** //porque recibira un valor

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*Para los LED\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

pinMode**(**LED1**,** OUTPUT**);** //Establecemos el pin 14 (D5) como salida

pinMode**(**LED2**,** OUTPUT**);** //Establecemos el pin 12 (D6) como salida

/\*\*\*\*\*\*Para el sensor DHT22 y el de humedad de suelo\*\*\*\*\*\*\*\*\*/

dht**.**begin**();** //Permite configurar el pin del sensor como INPUT\_PULLUP

pinMode**(**sensorHumedad**,** INPUT**);** //Definimos el pin del sensor como INPUT

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

Serial**.**begin**(**115200**);** //Se inicio la comunicación serial

setup\_wifi**();** //Llamamos a la función para conectarnos a nuestra red wifi

client**.**setServer**(**mqtt\_server**,** 1883**);** //Enviamos el servidor y nuestro puerto

client**.**setCallback**(**callback**);** //Establecemos el callback

**}**

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

void loop**()** **{**

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*Para el sensor DHT22\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

//Se lee la temperatura en Celcius y se le asigna a la variable tem

float tem **=** dht**.**readTemperature**();**

//Se lee la humedad relativa y se le asigna a la variable hum

float hum **=** dht**.**readHumidity**();** //Se lee la humedad relativa y se le asigna a hum

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*Para el sensor de humedad del suelo\*\*\*\*\*\*\*\*\*\*\*\*/

// Capturamos el valor del sensor de humedad de suelo en la variable hum\_suelo

// Realizamos un pequeño calculo para transformar la lectura analogica a un rango de 0-100

hum\_suelo **=** **(**100.00 **-** **((**analogRead**(**sensorHumedad**)** **\*** 100**)** **/** 1024.00**));**

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*Para el sensor ultrasonido\*\*\*\*\*\*\*\*\*\*\*\*/

//configuramos el trigger

digitalWrite**(**PinTrig**,** LOW**);** //Para generar un pulso limpio se pone a LOW por 4 us

delayMicroseconds**(**4**);**

digitalWrite**(**PinTrig**,** HIGH**);** //generamos un Trigger de 10 us

delayMicroseconds**(**10**);**

digitalWrite**(**PinTrig**,** LOW**);** //volvemos a poner en LOW

//calculamos el tiempo en microsegundos (us) que le toma hacer contacto con un objeto

tiempo **=** pulseIn**(**PinEcho**,** HIGH**);**

//La distancia se calcula dividiento el tiempo entre 58.3

//esto es porque se sabe que la velocidad del sonidos recorre

//1cm en 29.15us, y como solo nos interesa la mitad de esa distancia

//lo dividimos entre 2, por lo que con la regla de tres simple

//tenemos que la distancia=tiempo/(29.15\*2)

distancia **=** tiempo **/** 58.3**;**

lecturas**[**nActual**]** **=** distancia**;**

nActual**++;**

float promDistancia**;** //Valor promedio de 100 lecturas del sensor ultrasonido

//Calculamos un promedio de las distancias para tratar de evitar cualquier tipo de error

//en las lecturas del sensor ultrasonido

**if** **(**nActual **+** 1 **==** 100**)** **{** //si ya hemos llenado el arreglo, entonces ...

nActual **=** 0**;** //Reiniciamos el contador para volver a llenar el arreglo desde el indice 0

promDistancia **=** promedio**(**lecturas**);** //Almacenamos el promedio de las distancias

//Como lo que mostraremos en el Node-red seran el porcentaje de agua que hay en el recipiente

//entonces aplicamos unas simples operaciones matematicas para obtenerlo

porcentajeAgua **=** **(**1 **-** **((**promDistancia **-** distanciaLleno**)** **/** **(**distanciaVacio **-** distanciaLleno**)))** **\*** 100**;**

//Agregamos una validaciones frente a posibles movimientos o perturbaciones en el sensor ultrasonido

**if** **(**porcentajeAgua **<** 0**)** **{**

porcentajeAgua **=** 0**;**

**}** **else** **if** **(**porcentajeAgua **>** 100**)** **{**

porcentajeAgua **=** 100**;**

**}**

//Para el encendido del led 2

**if** **(**porcentajeAgua **<** 20**)** **{** //Si el porcentaje de agua del recipiente esta por debajo del 20%, entonces...

digitalWrite**(**LED2**,** HIGH**);** //encendemos el led2

**}** **else** **{**

digitalWrite**(**LED2**,** LOW**);** //apagamos el led2

**}**

**}**

delay**(**10**);** //esperamos 10 milisegundos

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

//Si nos desconectamos trateremos de volver a conetarnos

**if** **(!**client**.**connected**())** **{**

reconnect**();**//llamamos a la función reconnect

**}**

client**.**loop**();** //hacemos un loop

riego\_automatico**(**tem**,** hum**,** hum\_suelo**);** //Llamamos a la función del riego automatico

long now **=** millis**();** //Calculamos el tiempo actual

//Si el tiempo actual menos el del ultimo mensaje es mayor igual a 5000

//se publicaran los datos

**if** **(**now **-** lastMsg **>=** 5000**)** **{**

lastMsg **=** now**;** //actualizamos el momento en donde se recibe el ultimo mensaje

//La funcion dtostrf nos permite convertir una variable de tipo double o float en su

//representación ASCII y almacenarla en un arreglo de tipo char, que seran posteriormente publicadas,

//primera se coloca la variable que deseamos convertir, luego el ancho de esta, el numero de decimales

//y por ultimo donde queremos almacenarla

dtostrf**(**tem**,** 0**,** 2**,** dht22tem**);**

dtostrf**(**hum**,** 0**,** 2**,** dht22hum**);**

dtostrf**(**hum\_suelo**,** 0**,** 2**,** sensorhum**);**

dtostrf**(**porcentajeAgua**,** 0**,** 2**,** sensorultrasonic**);**

//Imprimimos los datos en el puerto serial

Serial**.**print**(**tem**);**

Serial**.**print**(**","**);**

Serial**.**print**(**hum**);**

Serial**.**print**(**","**);**

Serial**.**print**(**hum\_suelo**);**

Serial**.**print**(**","**);**

Serial**.**print**(**porcentajeAgua**);**

Serial**.**print**(**","**);**

Serial**.**print**(**estadoBomba**);**

Serial**.**println**();**

//Publicamos los datos

client**.**publish**(**"dht22-tem"**,** dht22tem**);** //publicamos la temperatura

client**.**publish**(**"dht22-hum"**,** dht22hum**);** //publicamos la humedad

client**.**publish**(**"sensor-hum\_suelo"**,** sensorhum**);** //publicamos la humedad del suelo

client**.**publish**(**"sensor-ultrasonido"**,** sensorultrasonic**);** //publicamos el porcentaje de agua del recipiente

delay**(**500**);** //esperamos 500 milisegundos

**}**

**}**

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

void riego\_automatico**(**float tem**,** float hum**,** float hum\_suelo**)** **{**

//Iniciamos con la bomba apagada

digitalWrite**(**bombaAgua**,** HIGH**);** //apagar bomba de agua

delay**(**100**);** //esperamos 100 milisegundos

**if** **((**tem **>** 30 **&&** hum **<** 30 **&&** hum\_suelo **<** 30 **)** **||** **(**hum\_suelo **<** 20 **))** **{**

//el LED1 se encendera por 5 segundos antes de activarse el riego y estara encendida hasta que termine el riego

digitalWrite**(**LED1**,** HIGH**);** //encendemos el led1

delay**(**5000**);**

digitalWrite**(**bombaAgua**,** LOW**);** //prender bomba de agua

estadoBomba **=** "1"**;** //Actualizamos los estados

delay**(**5000**);** //la bomba estara encendidad por 5 segundos

**}**

**else** **if** **(**mensaje **==** "control\_riego"**)** **{**

**if** **(**inform **==** "1"**)** **{**

digitalWrite**(**bombaAgua**,** LOW**);** //prender bomba de agua

digitalWrite**(**LED1**,** HIGH**);** //encendemos el led1 mientras la bomba este encendida

//Actualizamos los estados

estadoBomba **=** "1"**;**

delay**(**5000**);** //Activamos el riego por 5 segundos

**}**

**else** **{**

digitalWrite**(**bombaAgua**,** HIGH**);** //apagar bomba de agua

digitalWrite**(**LED1**,** LOW**);** //apagamos el led1

//Actualizamos los estados

estadoBomba **=** "0"**;**

**}**

**}**

**else** **{** //si no pasa nada de los anterior estara apagada

digitalWrite**(**bombaAgua**,** HIGH**);** //apagar bomba de agua

digitalWrite**(**LED1**,** LOW**);** //tambien apagamos el led1

//Actualizamos los estados

estadoBomba **=** "0"**;**

**}**

**}**

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

//Creamos una función para calcular el promedio de las lecturas en el arreglo

float promedio**(**float a**[])** **{**

float total **=** 0.0**;**

**for** **(**int i **=** 0**;** i **<** nLecturas**;** i**++)** **{**

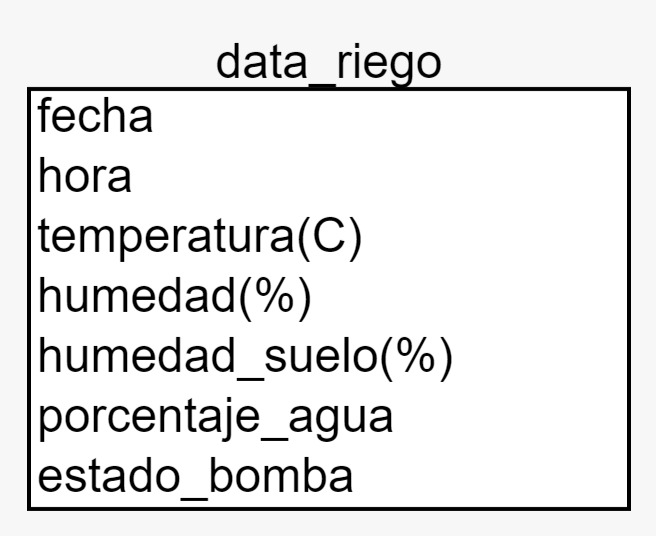
total **+=** a**[**i**];**

**}**

**return** total **/** nLecturas**;**

**}**

## Estructuración de los datos



## Imágenes del sistema de riego



