Informe Laboratorio IOT

Daniel Arturo Rincón Aldana

[danielarturo@unisangil.edu.co](mailto:danielarturo@unisangil.edu.co)

David Eduardo Torres Hidalgo

[davidtorres@unisangil.edu.co](mailto:davidtorres@unisangil.edu.co)

Brayan Arturo Castro Pinzón

[brayanarturo@unisangil.edu.co](mailto:brayanarturo@unisangil.edu.co)

RESUMEN – MQTT son las siglas MQ Telemetry Transport, aunque en primer lugar fue conocido como Message Queing Telemetry Transport. Es un protocolo de comunicación M2M (machine-to-machine) de tipo message queue, está basado en la pila TCP/IP como base para la comunicación. En el caso de MQTT cada conexión se mantiene abierta y se "reutiliza" en cada comunicación. Es una diferencia, por ejemplo, a una petición HTTP 1.0 donde cada transmisión se realiza a través de conexión.

MQTT fue creado por el Dr. Andy Stanford-Clark de IBM y Arlen Nipper de Arcom (ahora Eurotech) en 1999 como un mecanismo para conectar dispositivos empleados en la industria petrolera, aunque inicialmente era un formato propietario, en 2010 fue liberado y pasó a ser un estándar en 2014 según la OASIS (Organization for the Advancement of Structured Information Standards).

.

**Índice de Términos –** MQTT, Comunicación, Conexión, Petición.

ABSTRACT-MQTT stands for MQ Telemetry Transport, although it was first known as Message Queing Telemetry Transport. It is an M2M (machine-to-machine) communication protocol of the message queue type, it is based on the TCP / IP stack as the basis for communication. In the case of MQTT each connection is kept open and "reused" in each communication. It is a difference, for example, to an HTTP 1.0 request where each transmission is done through connection.

MQTT was created by Dr. Andy Stanford-Clark of IBM and Arlen Nipper of Arcom (now Eurotech) in 1999 as a mechanism to connect devices used in the oil industry, although initially it was a proprietary format, in 2010 it was released and passed to be a standard in 2014 according to the OASIS (Organization for the Advancement of Structured Information Standards).

**Terms Index -** MQTT, Communication, Connection, Request.

# **I. INTRODUCCION**

El Internet de las cosas o Internet of Things (que en el resto del artículo referiremos simplemente como IoT, por sus siglas en inglés), es la tecnología que permite la interconexión de dispositivos físicos, vehículos, edificios y otros elementos integrados con electrónica, software, sensores, actuadores y redes de conectividad que permiten a estos objetos recopilar e intercambiar datos para su posterior análisis, a diferencia del procesamiento y control de la señal digital tradicional que consumen demasiada potencia en los nodos y demasiado ancho de banda en la red al procesar series de tiempo periódicas, la tecnología IoT permite realizar muestreo por evento o muestreo aperiódico reduciendo así el consumo energético.

Por otro lado, los sistemas de control y monitoreo en tiempo real son importantes en múltiples áreas y aplicaciones, gracias a que otorgan la habilidad de poder monitorear la eficiencia del sistema analizado, además de poder diagnosticar errores y fallas presentes en estos. Estas fallas se ven reflejadas en las múltiples herramientas disponibles en los sistemas de monitoreo en tiempo real, a través de gráficas de las variables monitoreadas. Dentro de los sistemas de control, el control domótico permite automatizar sistemas eléctricos y electrónicos ejecutando rutinas programadas para así mejorar la eficiencia energética de estos sistemas y proveer facilidad de manejo al usuario final.

Combinando características de monitoreo y control domótico con la tecnología IoT, se puede obtener un sistema de análisis de eficiencia y control de cargas eléctricas y electrónicas para poder ver así el efecto de incidencia sobre el sistema analizado y el consumo total energético, debido a las restricciones en potencia y ancho de banda, el IoT permite la computación distribuida de los eventos generados por los sensores, Si bien los procesadores relativamente pequeños de los sensores pueden realizar un procesamiento útil en muchas secuencias de datos, el reconocimiento de eventos de interés, mediante el procesamiento externo y envío de datos utilizando un protocolo ligero, reduce la cantidad de ancho de banda de la red consumida y también reduce el consumo de energía ya que la comunicación inalámbrica que requiere grandes cantidades de energía.

En el proceso de aprendizaje y uso de MQTT, una práctica herramienta de cliente puede facilitar enormemente a los usuarios la exploración de las características de MQTT y la depuración de componentes funcionales. Los desarrolladores de todo el mundo han desarrollado muchas herramientas de prueba de clientes para el protocolo MQTT en diferentes sistemas operativos y plataformas operativas, existen muchos tipos de estas herramientas cliente, con diferentes funciones y diferentes niveles de calidad, por lo que, para principiantes e incluso expertos en MQTT, cómo elegir una herramienta cliente MQTT adecuada es un problema difícil, este artículo recopilará y organizará tanto como sea posible, y hará una evaluación integral de varias herramientas de cliente MQTT en el mercado para referencia de los lectores.

# **OBJETIVOS**

# **II. OBJETIVO GENERAL**

* Realizar una conexión MQTT entre el Bróker y el microcontrolador

# **III. OBJETIVOS ESPESIFICOS**

* Comprender el funcionamiento de las herramientas MQTT Y Bróker.
* Implementar los sensores DTH11, HC-SR04, ESP 32, para la obtención y envió de datos.
* Elaborar la programación para la Conexión MQTT, Bróker – Cliente para el funcionamiento de los sensores mencionados con anterioridad.
* Mostrar las caracteristicas y funcionalidades del protocolo MQTT en la implementación de la conexión

**IV. CODIGO PROGRAMA**

**# incluir < Arduino.h >**

**# incluye < WiFi.h >**

**# include < PubSubClient.h >**

**# include < ESP32Servo.h > // incluye libreria de Servo para la placa**

**# include " DHT.h " // incluyo la libretia para el sensor de temperatura**

**# include " ESP32\_MailClient.h " // incluyo la libreria para enviar correos**

**# definir DHTPIN 15**

**# definir DHTTYPE DHT11**

**DHT DHT (DHTPIN, DHTTYPE); // creo objeto para el sensor de temperatura**

**Servo servo1; // crea objeto para el servo**

**// defino variables del sensor ultrasónico**

**const int trigPin = 2 ; // son las entradas para el sensor de distancia**

**const int echoPin = 5 ;**

**// defino variables del servomotor**

**int PINSERVO = 26 ; // pin 34 conectado a señal del servo**

**int PULSOMIN = 900 ; // pulso minimo en microsegundos**

**int PULSOMAX = 2000 ; // pulso maximo en microsegundos**

**// defino demas variables**

**larga duración;**

**int D\_ant, Distancia, bandera, mnum, BANDERA;**

**int LED = 13 ; // para prender el bombillo 1**

**int LED1 = 12 ; // para prender el bombillo 2**

**int VENT = 14 ; // para prender el ventilador**

**int LED\_MAIL = 18 ; // para prender el led indicador que se envio el mensaje**

**int BOTON = 0 ; // una bandera**

**int SENSOR = 4 ; // para leer la apertura de la puerta**

**int mnuman = 0 ; // para almacenas el valor de temperatura naterior en entero**

**int cont = 500 ;**

**int BANDERA1 = 0 ;**

**String stringdistance, str, tem, hum, mens, PUERTA, EMAIL;**

**flotar t, h, h\_ant, t\_ant;**

**SMTPData datosSMTP;**

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

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

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

**const char \* mqtt\_server = " ioticos.org " ;**

**const int mqtt\_port = 1883 ;**

**const char \* mqtt\_user = " ObR3tRVx5nghNer " ;**

**const char \* mqtt\_pass = " ZqpHVjEV84O48RS " ;**

**const char \* root\_topic\_subscribe = " hF39Jx1E462loPz / input " ;**

**const char \* root\_topic\_publish = " hF39Jx1E462loPz / salida " ;**

**const char \* root\_topic\_tem = " hF39Jx1E462loPz / tem " ;**

**const char \* root\_topic\_hum = " hF39Jx1E462loPz / hum " ;**

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

**// \*\*\*\*\*\*\*\*\*\*\* WIFICONFIG \*\*\*\*\*\*\*\*\*\*\*\*\*\*\***

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

**const char \* ssid = " BOCADITOS DE AMOUR " ;**

**const char \* contraseña = " Kinkbmx. " ;**

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

**// \*\*\*\*\*\*\*\*\*\*\* GLOBALES \*\*\*\*\*\*\*\*\*\*\*\*\*\*\***

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

**WiFiClient espClient;**

**Cliente PubSubClient (espClient);**

**char msg [ 76 ];**

**char msgtem [ 25 ];**

**char msghum [ 25 ];**

**// cuenta larga = 0;**

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

**// \*\* FUNCIONES \*\*\***

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

**devolución de llamada void ( char \* topic, byte \* payload, unsigned int length);**

**anular reconectar ();**

**void setup\_wifi ();**

**configuración vacía ()**

**{**

**De serie. comenzar ( 9600 );**

**setup\_wifi ();**

**cliente. setServer (servidor\_mqtt, puerto\_mqtt);**

**cliente. setCallback (devolución de llamada);**

**servo1. adjuntar (PINSERVO, PULSOMIN, PULSOMAX); // inicializacion de servo**

**pinMode (trigPin, SALIDA); // defino como salidad el pin tring del ultrasoido**

**pinMode (echoPin, ENTRADA); // defino como entrada el pin echo del ultrasoido**

**pinMode (LED, SALIDA);**

**pinMode (LED1, SALIDA);**

**pinMode (VENTILACIÓN, SALIDA);**

**pinMode (LED\_MAIL, SALIDA);**

**pinMode (SENSOR, ENTRADA); // Pulsador o sensor como entrada**

**escritura digital (LED, ALTA); // les doy un alto iniciando para que no se prendan los relay ya que prenden con ceros**

**escritura digital (LED1, ALTA);**

**escritura digital (VENTILACIÓN, ALTA);**

**digitalWrite (LED\_MAIL, BAJO);**

**dht. comenzar (); // inicio el modulo dht11**

**}**

**bucle vacío ()**

**{**

**if (! client. connected ()) // si el cliente no se conecta entra**

**{**

**reconectar ();**

**}**

**si (cliente. conectado ())**

**{**

**// limpia los pines de trigpin**

**digitalWrite (trigPin, LOW);**

**delayMicroseconds ( 10 );**

**h = dht. readHumidity (); // leo la humedad del sensor y lo guardo en h**

**t = dht. readTemperature (); // leo la temperatura del sensor y lo guardo en t**

**// coloca el trigpin en alto durante 10 microsegundos**

**digitalWrite (trigPin, HIGH);**

**delayMicroseconds ( 10 );**

**digitalWrite (trigPin, LOW);**

**// lee el pin echo y debuelvo el tiempo de ida y vuelta de la onda en microsegundos**

**duración = pulseIn (echoPin, HIGH);**

**// Calculo la distancio con esta ecuacion**

**Distancia = duración \* 0.034 / 2 ;**

**if (BOTON == 1 ) PUERTA = " ABIERTA " ; // son para escribir si esta abierta o cerrada la puerta segun el caso**

**más PUERTA = " CERRADA " ;**

**str = " Distancia: " + Cadena (Distancia) + " Cm " + " Humedad: " + h + " % Temperatura: " + t + " ° C puerta: " + PUERTA;**

**tem = " Temperatura: " + Cadena (t) + " ° C " ;**

**hum = " Humedad: " + Cadena (h) + " % " ;**

**// muestro en pantalla las variables**

**if (D\_ant! = Distance || h\_ant! = h || t\_ant! = t) // si hay algun cambio muestre en pantalla**

**{**

**str. toCharArray ( mensaje , 76 ); // convierta str en arreclo char con 57 carateres**

**cliente. publicar (root\_topic\_publish, msg);**

**}**

**if (cont> 500 ) // esto es para masomenos contar 25segundos para mostrar el valor de temperatura y humedad**

**{**

**tem. toCharArray (msgtem, 25 );**

**cliente. publicar (root\_topic\_tem, msgtem);**

**tararear. toCharArray (msghum, 25 );**

**cliente. publicar (root\_topic\_hum, msghum);**

**cont = 0 ;**

**}**

**else cont ++;**

**D\_ant = Distancia; // guarda el cmabio anterior para la funcion si cambia la variables muestra**

**h\_ant = h;**

**t\_ant = t;**

**BOTON = digitalRead (SENSOR); // lee si esta abierta la puerta**

**si (BOTON == 1 )**

**{**

**correo (); // se llama a correo para decir que esta abierta la puerta**

**}**

**casos ();**

**}**

**cliente. bucle ();**

**}**

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

**// \*\*\* CONEXION WIFI \*\*\***

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

**vacío setup\_wifi ()**

**{**

**retraso ( 10 );**

**// Nos conectamos a nuestra red Wifi**

**De serie. println ();**

**De serie. print ( " Conectando un ssid: " );**

**De serie. println (ssid);**

**Wifi. comenzar (ssid, contraseña);**

**mientras (WiFi. status ()! = WL\_CONNECTED)**

**{**

**retraso ( 500 );**

**De serie. imprimir ( " . " );**

**}**

**De serie. println ( " " );**

**De serie. println ( " Conectado un WiFi rojo! " );**

**De serie. println ( " Dirección IP: " );**

**De serie. println (WiFi. localIP ());**

**}**

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

**// \*\*\* CONEXIÓN MQTT \*\*\***

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

**anular reconectar ()**

**{**

**mientras (! cliente. conectado ())**

**{**

**De serie. print ( " Intentando conexión Mqtt ... " );**

**// Creamos un ID de cliente**

**String clientId = " IOTICOS\_H\_W\_ " ;**

**clientId + = String ( aleatorio ( 0xffff ), HEX);**

**// Intentamos conectar**

**if (client. connect (clientId. c\_str (), mqtt\_user, mqtt\_pass))**

**{**

**De serie. println ( " Conectado! " );**

**// Nos suscribimos**

**if (cliente. subscribe (root\_topic\_subscribe))**

**{**

**De serie. println ( " Suscripcion ok " );**

**}**

**demás**

**{**

**De serie. println ( " fallo Suscripciión " );**

**}**

**}**

**demás**

**{**

**De serie. print ( " falló :( con error -> " );**

**De serie. imprimir (cliente. estado ());**

**De serie. println ( " Intentamos de nuevo en 5 segundos " );**

**retraso ( 5000 );**

**}**

**}**

**}**

**// \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* desifra la informacion del brouquer**

**// \*\*\* CALLBACK \*\*\* en formato char**

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

**devolución de llamada vacía ( char \* tema, byte \* carga útil, longitud int sin firmar )**

**{**

**Cadena entrante = " " ;**

**De serie. print ( " Mensaje recibido desde: " );**

**De serie. imprimir (tema);**

**De serie. println ( " " );**

**para ( int i = 0 ; i <longitud; i ++)**

**{**

**entrante + = ( char ) payload [i];**

**}**

**mens = entrante; // guardo en una variable para poder usarlo como lectura**

**entrante. recortar ();**

**De serie. println ( " Mensaje: " + entrante);**

**casos (); // llama la funcion y según la orden ejecuta**

**}**

**void casos () // funcion "funcion casos"**

**{**

**// estos if son para prender y apagar el bombillo de la abitacion # 1**

**if (mens == " a0 " ) // if (mens == "cuarto1off")**

**{**

**escritura digital (LED, ALTA);**

**// Serial.println ("entro a apagar cuarto1");**

**}**

**if (mens == " a1 " ) // if (mens == "cuarto1on")**

**{**

**escritura digital (LED, BAJA);**

**// Serial.println ("entro a prender cuarto1");**

**}**

**// estos if son para prender y apagar el bombillo de la abitacion # 2**

**if (mens == " b0 " ) // if (mens == "cuarto2off")**

**{**

**escritura digital (LED1, ALTA);**

**// Serial.println ("entro a apagar cuarto2");**

**}**

**if (mens == " b1 " ) // if (mens == "cuarto2on")**

**{**

**escritura digital (LED1, BAJA);**

**// Serial.println ("entro a prender cuarto2");**

**}**

**// estos if son para prender y apagar el ventilador**

**if (mens == " c0 " ) // if (mens == "ventiladoroff")**

**{**

**escritura digital (VENTILACIÓN, ALTA);**

**// Serial.println ("entro a apagar ventilador");**

**}**

**if (mens == " c1 " ) // if (mens == "ventiladoron")**

**{**

**escritura digital (VENTILACIÓN, BAJA);**

**// Serial.println ("entro a prender ventilador");**

**}**

**// estos if son para cerrar o abrir la puerta principal de la casa**

**if (mens == " d0 " && BANDERA1> 0 ) // if (mens == "cerrar")**

**{**

**servo1. escribir ( 0 ); // ubica el servo a 0 grados**

**retraso ( 200 );**

**BANDERA1 = 0 ;**

**De serie. println ( " entro a cerrar puerta " );**

**}**

**if (mens == " d1 " && BANDERA1 < 1 ) // if (mens == "abrir")**

**{**

**servo1. escribir ( 180 ); // ubica el servo a 0 grados**

**// correo ();**

**BANDERA1 ++; // reseteo la bandera**

**De serie. println ( " entro a abrir puerta " );**

**retraso ( 200 );**

**// Serial.println ("entro a abrir puerta");**

**}**

**// este if es para cuando el cliente desea poner una temeratura para que se prende el ventilador**

**if (mens [ 0 ] == ' t ' && mens [ 1 ] == ' e ' && mens [ 2 ] == ' m ' && mens [ 3 ] == ' : ' )**

**{**

**mnum = ((mens [ 4 ] - 48 ) \* 10 ) + mens [ 5 ] - 48 ; // guardo el numero para usarlo en la temperatura**

**De serie. println ( " entro al tem: " );**

**if (t> = mnum) // si la tempratura es mayor o igual a la designada por el usuario entro**

**{**

**escritura digital (VENTILACIÓN, BAJA); // prendo el ventilaror (trabaja con ceros)**

**// Serial.println ("entro al mayor que");**

**retraso ( 1000 );**

**}**

**de lo contrario digitalWrite (VENT, HIGH);**

**if (mnum! = mnuman) // si cambio la tempratura**

**{**

**BANDERA = 1 ; // uso esta bandera para indicar que es para la temperatura**

**correo ();**

**BANDERA = 0 ; // reseteo la bandera**

**}**

**mnuman = mnum; // para que no entre si no cuando la temperatura cambie**

**}**

**}**

**correo vacío ()**

**{**

**De serie. print ( " Iniciando correo !!! " );**

**retraso ( 1000 );**

**CORREO ELECTRÓNICO = " SE ACABA DE ABRIR LA PUERTA " ;**

**if (BANDERA == 1 ) EMAIL = " CAMBIO LA TEMPERATURA " ; // este if es para cuando se esta cambiando la temperatura y no abriendo la puerta**

**digitalWrite (LED\_MAIL, ALTO); // prendo el led indicando que se esta enviendo un email**

**datosSMTP. setLogin ( " smtp.gmail.com " , 465 , " labiot2021@gmail.com " , " laboratorio2021 " ); // Configuración del servidor de correo electrónico SMTP, host, puerto, cuenta y contraseña**

**datosSMTP. setSender ( " DANIEL\_ESP32 " , " labiot2021@gmail.com " ); // coloca el nombre del que envia el mensaje y el correo electrónico**

**datosSMTP. setPriority ( " Alto " ); // Establezca la prioridad o importancia del correo electrónico Alta, Normal, Baja o 1 a 5 (1 es el más alto)**

**datosSMTP. setSubject ( " ALERTA DE SEGURIDAD " ); // Establecer el asunto**

**datosSMTP. setMessage (CORREO ELECTRÓNICO, falso ); // escribe el contenido del correo**

**datosSMTP. addRecipient ( " danielrincon@unisangil.edu.co " ); // aqui se envia el correo al destinatario**

**if (! MailClient. sendMail (datosSMTP)) // si tiene datos comience a enviar correo electrónico.**

**De serie. println ( " Error enviando el correo, " + MailClient. smtpErrorReason ());**

**datosSMTP. vacío (); // Borrar todos los datos del objeto datosSMTP para liberar memoria**

**retraso ( 9000 ); // esperao 9min a que se envie el mensaje**

**digitalWrite (LED\_MAIL, BAJO); // apago lideró despues de enviar el mensaje**

**BANDERA == 0 ;**

**}**

**V. PROCEDIMIENTO**

* **Agregamos una librería en Arduino para que pueda haber conexión entre el microcontrolador y la programación en Arduino**
* **Incluimos la librería “WIFI” en Arduino, que nos ayuda para la comunicación a la Internet por medio de los protocolos WAP y WAP2**
* **Incluimos la librería “PUBSUBCLIENT”, para que nuestra placa de Arduino se comporte como un cliente MQTT, esto nos permite publicar y suscribirnos a los tópicos.**
* **Incluimos la librería “ESP32SERVO”, que nos permite por medio del microcontrolador manipular el servomotor internamente con PWM**
* **Incluimos la librería “DHT”, que nos permite monitorear por medio del sensor de temperatura y humedad del ambiente**
* **Defino DHTPIN por el puerto 15, para sensar la temperatura y humedad**
* **Defino DHT11 para el sensor que se va utilizar**
* **Se crea un DHT DHT (DHTPIN, DHTTYPE) y Servo servo1 para el sensor de temperatura, que permite decidir con que tipo de sensor se va a implementar**
* **Defino las constantes enteras del sensor ultrasónico donde va a determinar la distancia “trigPin por el puerto 2 y echopin por el puerto 5”**
* **Defino las variables enteras para el servomotor, que permiten el movimiento mediante PWM “PINSERVO por el puerto 26 (que permite recibir el comando de la posicion), PULSOMIN=900 y PULSOMAX=2000 (estos permiten controlar el Angulo máximo y mínimo de giro)**
* **Defino unas variables enteras y una long, en el caso de la variable long que trabaja con 8 bits permite almacenar datos grandes de información, esta variables la denominamos “DURATION”, permite guardar el tiempo de ida y vuelta del sonido emitido por el sensor, las variable “D\_ant, H\_ant, T\_ant”, van almacenar los varoles anteriores de la temperatura, distancia y movimiento, las variables “bandera y BANDERA”, son banderas que se activan con unos procesos determinados, variable “MNUM” permite almacenar la temperatura que desea el usuario, variable “LED1 Y LED” están ubicados en los puertos 12 y 13 respectivamente, estos permiten controlar los focos, variable “VENT” ubicado en el puerto 14, permite controlar el encendido y apagado del sistema de enfriamiento, variable “LED\_MAIL” indica si se está enviando información al correo, variable “BOTON” lo usamos para escribir el estado de la puerta y se inicializa en 0, variable “SENSOR” esta permite leer el estado de la puerta y se encuentra ubicado en el puerto 4, variable “MNUMAN” la inicializamos en 0, permite almacenar los datos pasados recogidos por la variable “MNUM”, variable “CONT” lo inicializamos en 500, esta permite mostrar las variables de temperatura y humedad, variable “BANDERA1” lo inicializamos en 0, esta permite averiguar en que estado se encuentra la puerta para controlar el servomotor.**
* **Se crean las variables string, las cuales son cadenas de caracteres, variable “str, temp, hum” permite almacenar los datos obtenidos por todas la variables, “MENS” se utiliza para almacenar el “string” en otra variable, y poder leerlo, variable “PUERTA Y EMAIL” permiten almacenar en caracteres en que estado se encuentra la puerta.**
* **Se crean variables “float” que permiten almacenar números decimales, variable “t, h” se utiliza para almacenar los datos de humedad y temperatura.**
* **Se crea un “datosSMTP”, este nos permite activar la función de enviar mensajes o comunicarse vía correo electrónico.**
* **Definimos unas constante de tipo char y entero, para configurar el MQTT.**
* **Definimos unas constantes de tipo char y entero, para configurar el WIFI**
* **Se crea un “espClient”, que permite ser la instancia de cliente parcialmente inicializada, se crean tres variables tipo char “MSG, MSGTEMP, MSGHUM” que permiten almacenar el mensaje que se va a enviar a los topicos**
* **Llamo a una función callback, que permite leer los datos ingresados por el cliente**
* **Llamo a la función reconnect, que permite conectarnos al MQTT.**
* **Llamo a la función setup\_wifi, que permite conectarnos a la red inalámbrica**
* **En setup, inicializo el puerto serial, el servomotor, el modulo DHT11, y definimos unos puertos de entrada y salida, y le ingresamos variables de “alto y bajo” a unas variables (bombillos, ventilador, etc.)**