Instituto Tecnológico y de Estudios Superiores de Monterrey Campus Puebla

Implementación de internet de las cosas (Gpo.2)

Docentes:

Cesar Torres Huitzil

Alba Adriana Romero García

Carlos Axel García Vega

Alumno:

Francisco Rocha Juárez / A01730560

Reporte de situación problema

Fecha:

29/11/2021

Acercamiento:

Las IoT son muy importantes e innovadoras en el día a día, muchas empresas las han empezado a implementar en sus líneas de producción ya que les beneficia de una manera muy satisfactoria, recortes en tiempos de producción, empaquetado y entrega.
El objetivo de este reto o situación problema se trata de generar una solución innovadora basándonos en las IoT esto para generar una solución a una necesidad actual esto en el contexto de entornos inteligentes esto entre una interfaz computacional con el mundo físico, para así obtener la manipulación, adquisición, la organización y el almacenamiento de los datos, para que estas puedan generar el análisis y después la toma de decisiones para poder tomar acciones inmediatas y así generar beneficios a la calidad de vida de la población.
La implementación de esta propuesta estará basada en dar solución a los problemas de control de calidad que se tienen en los productos no perecederos en las redes logísticas a lo largo de la cadena de suministros, es lo que motivó esta solución. Un dispositivo que pueda monitorear su humedad y temperatura en todo momento para poder tener un control de calidad óptimo y crear una red logística inteligente a la que los datos se puedan acceder en tiempo real y procesar.
Elegimos este tipo de empresa ya que es muy fácil de poder adoptar las IoT, además de que es fácil de poder implementar los conceptos vistos y poder dar una solución a los objetivos que se nos presentan en esta situación problema.

Introducción:

Sabemos que hoy en día las empresas empiezan por innovar en cuanto al manejo de la información, hoy en día, esa innovación se maneja mediante el uso de las nuevas tecnologías que se presentan, una de esas nuevas tecnologías es el Internet de las Cosas (IoT Internet of Things en inglés). La función principal del IoT es conectar todas las cosas mediante el internet, esto para obtener información de cada una de ellas, además de generar valor a los usuarios y a los negocios.

Según estudios, se estima que en los próximos años, todos y cada uno de nosotros tendremos entre 5 y 7 dispositivos conectados a internet (entre estos pueden considerarse laptops, computadoras de escritorio, teléfonos inteligentes, tablets, relojes inteligentes, etc.) igual

podemos incluir electrodomésticos, alarmas, incluso los autos. Si se sigue con esta tendencia, podemos esperar que se llegue a la cantidad de 50 billones de dispositivos conectados.

El IoT nos puede ser de gran ayuda para situaciones que pueden generar un alto impacto social, por dar un ejemplo; en la situación actual en la que nos encontramos, al aplicar el IoT podemos obtener información de los objetos que usamos y desarrollar servicios de inocuidad para los ciudadanos.

En este caso vamos a implementar una solución con IoT que nos permita reforzar las redes logísticas en la exportación de productos, perecederos, siendo en este caso el aguacate como ejemplo.

También tomando en cuenta que esto responde a una necesidad acorde a los **ODS**, objetivos de desarrollo sostenible de la ONU. Siendo el número 9 el que atendemos en esta situación problema **industria, innovación e infraestructuras,** siguiendo la meta **9.4** y **9.c** que dicen lo siguiente:

9.4 De aquí a 2030, modernizar la infraestructura y reconvertir las industrias para que sean sostenibles, utilizando los recursos con mayor eficacia y promoviendo la adopción de tecnologías y procesos industriales limpios y ambientalmente racionales, y logrando que todos los países tomen medidas de acuerdo con sus capacidades respectivas

9.c Aumentar significativamente el acceso a la tecnología de la información y las comunicaciones y esforzarse por proporcionar acceso universal y asequible a Internet en los países menos adelantados de aquí a 2020

La logística es indispensable cuando hablamos de cadenas de suministro, ya que sin esta nos sería imposible organizar todo el camino que debe seguir nuestra materia prima o producto para poder disfrutarla finalmente como clientes.

Todas las cadenas de suministro son diferentes, muchos factores se ven involucrados, como la distancia, los lugares a los que se quiera llegar, el tipo de cliente, el giro del producto, sus dimensiones, entre otras cosas. En este caso nosotros nos estamos centrando en los productos perecederos, como lo es el aguacate, es por ello que la logística durante la cadena de suministros se tiene que aplicar de manera diferente a los otros productos.

Con el aguacate si nosotros tardamos mucho desde que se corta el fruto hasta la entrega con el cliente, esto podría ser inservible y podría perderse el valor que durante toda la cadena fue generado. Por esto necesitamos de una logística eficiente y eficaz, pero que al mismo tiempo los precios sean asequibles y no se desborden.

En la actualidad contamos con mucha tecnología, esta tecnología también nos es de mucha utilidad en las cadenas de suministro como lo son los artefactos GPS, chips RFID, robots mecánicos para manipular la mercancía, softwares de optimización, simuladores logísticos entre otros, en esta solución nosotros ocuparemos sensores de humedad y temperatura conectados a un servidor en la nube en la plataforma ThingSpeak, enviando información mediante el protocolo MQTT con un ESP32, esto escrito en código en Arduino IDE. Esto va a servir para poder saber la temperatura y humedad del aguacate en todo momento y en caso de no ser lo que se necesita, enviar alertas, poder tomar decisiones al respecto y evitar pérdidas de producto que se traducen en pérdidas económicas.

Si nosotros sabemos ocupar esta tecnología de una manera adecuada podríamos reducir muchos de los costos, saber con exactitud las variables que tenemos involucradas en el producto, saber el tiempo exacto en que va a llegar el producto a los almacenes, identificar en donde se encuentra cada producto en cada etapa de la cadena de suministro, crear casos hipotéticos y saber que acciones tomar en diferentes circunstancias, etc…

Es por ello que en este documento se discutirán las formas de fortalecer la logística en el caso de un producto perecedero como lo es el aguacate. El uso e implementación de tecnología puede facilitar enormemente el trabajo y darnos una mejor comprensión. Al mismo tiempo podríamos requerir de menos personal laborando y aprovechar las oportunidades tecnológicas que se nos ofrecen.

Marco teórico:

- **IoT:**

"La definición de IoT podría ser la agrupación y conexión de dispositivos y objetos a través de una red (ya sea privada o Internet, la red de redes) en la que todos pueden ser visibles e interactuar. El tipo de artículo o dispositivo puede ser cualquier cosa, desde sensores y dispositivos mecánicos hasta artículos cotidianos como un refrigerador, zapatos o ropa. Todo lo que pueda imaginar podría estar conectado a Internet e interactuar sin intervención

humana, por lo que el objetivo es la interacción de máquina a máquina, o la llamada interacción M2M (máquina a máquina) o dispositivos M2M." *(Deloitte, 2021)*

- **Sensores:**

"Un sensor es un dispositivo que detecta el cambio en el entorno y responde a alguna salida en el otro sistema. Un sensor convierte un fenómeno físico en un voltaje analógico medible (o, a veces, una señal digital) convertido en una pantalla legible para humanos o transmitida para lectura o procesamiento adicional.

Dependiendo del tipo de sensor, su salida eléctrica puede ser un voltaje, corriente, resistencia u otro atributo eléctrico que varía con el tiempo. Algunos sensores están disponibles con salidas digitales, por lo que generan una serie de bytes de datos escalados o no escalados. La salida de estos sensores analógicos generalmente está conectada a la entrada de un acondicionador de señal" (Dewesoft, 2021)

En el contexto de este reto se ocupa un sensor de temperatura, para ser específico el sensor DHT11.

- **Actuadores:**

"Un actuador es un dispositivo con la capacidad de generar una fuerza que ejerce un cambio de posición, velocidad o estado de algún tipo sobre un elemento mecánico, a partir de la transformación de energía.
Por lo regular, los actuadores se clasifican en dos grandes grupos:

1. *Por el tipo de energía utilizada: actuador neumático, hidráulico y eléctrico.*
2. *Por el tipo de movimiento que generan: actuador lineal y rotatorio. " (Ramírez, 2014)*

La ejecución del actuador depende de las mediciones realizadas por el sensor, está condicionado, mientras el sensor sensa el ambiente para la toma de decisiones el actuador se ejecuta para producir un cambio de estado en el ambiente. En el contexto del reto el sensor de temperatura cuando mida una temperatura anormal va a poner en funcionamiento los ventiladores o refrigeración, esto con fines prácticos va a ser simulado con un led.

- **Arquitecturas para internet de las cosas:**

"La arquitectura de tres capas define la idea principal de Internet de las cosas, pero no es suficiente para la investigación sobre IoT porque la investigación a menudo se centra en aspectos más finos de Internet de las cosas. Por eso, tenemos muchas más arquitecturas en capas propuestas en la literatura. Una es la arquitectura de cinco capas, que además incluye las capas de procesamiento y de negocio [3–6]. Las cinco capas son percepción, transporte, procesamiento, aplicación y capas comerciales (consulte la Figura 1). El papel de las capas de percepción y aplicación es el mismo que el de la arquitectura con tres capas. Delineamos la función de las tres capas restantes.

(i) La capa de transporte transfiere los datos del sensor desde la capa de percepción a la capa de procesamiento y viceversa a través de redes como inalámbricas, 3G, LAN, Bluetooth, RFID y NFC.

(ii) La capa de procesamiento también se conoce como capa de middleware. Almacena, analiza y procesa grandes cantidades de datos que provienen de la capa de transporte. Puede administrar y proporcionar un conjunto diverso de servicios a las capas inferiores. Emplea muchas tecnologías como bases de datos, computación en la nube y módulos de procesamiento de big data.

(iii) La capa empresarial gestiona todo el sistema de IoT, incluidas las aplicaciones, los modelos de negocio y beneficios, y la privacidad de los usuarios.

La arquitectura más básica es una arquitectura de tres capas. Se introdujo en las primeras etapas de la investigación en esta área. Tiene tres capas, a saber, las capas de percepción, red y aplicación.

(i) La capa de percepción es la capa física, que tiene sensores para detectar y recopilar información sobre el medio ambiente. Detecta algunos parámetros físicos o identifica otros objetos inteligentes en el entorno.

(ii) La capa de red es responsable de conectarse a otras cosas inteligentes, dispositivos de red y servidores. Sus funciones también se utilizan para transmitir y procesar datos de sensores.

(iii) La capa de aplicación es responsable de brindar servicios específicos de la aplicación al usuario. Define varias aplicaciones en las que se puede implementar Internet de las cosas, por ejemplo, hogares inteligentes, ciudades inteligentes y salud inteligente." (Sethi, 2017)

- **Arquitecturas de Nube de Iot:**

"En una arquitectura de nube de IoT, los datos fluyen a través de muchas capas. Aunque la configuración real de la plataforma en la nube puede diferir entre organizaciones, todas estas soluciones logran el mismo objetivo básico.

Esto incluye permitir que diferentes dispositivos se conecten a la red, procesando los datos y utilizando los conocimientos adquiridos para la automatización. Una parte importante del procesamiento de datos se lleva a cabo en la capa de informes y bases de datos en la nube.

Dispositivos de IoT
El dispositivo se integra con sensores o actuadores y establece una conexión con el Middleware de integración de IoT.

Middleware de integración de IoT
El Middleware de integración de IoT es una capa de integración para varios dispositivos cuando se conectan a la nube. Recibe datos de los dispositivos conectados, los procesa y transmite esta información a las aplicaciones posteriores. El procesamiento de datos puede incluir la evaluación de reglas de condición-acción y el despliegue de comandos a los sensores del dispositivo en base a la evaluación.

Si el dispositivo admite una tecnología de comunicación adecuada (WiFi o IP sobre Ethernet), un protocolo de transporte como MQTT o HTTP y un formato de carga útil compatible, puede transmitir datos directamente al Middleware de integración.

Servidores en la nube
Los servidores son la parte más importante de la nube de IoT, ya que son necesarios para brindar servicios comerciales a los clientes.

Bases de datos
Según los requisitos comerciales para el almacenamiento y procesamiento de datos, las bases de datos SQL y No SQL se pueden configurar en la nube de IoT.

Aplicaciones posteriores / Herramientas de BI
Los servidores en la nube están conectados a aplicaciones de terceros, aplicaciones móviles / web o herramientas de inteligencia empresarial a través de puntos finales de API REST."

- **Estándares y protocolos de comunicación:**

"En el ámbito informático y de telecomunicaciones, los protocolos IoT son un conjunto de normas y reglas que permiten a dos entidades entenderse e intercambiar información, facilitando la comunicación Machine2Machine (M2M).

En otras palabras, los protocolos IoT son a la comunicación entre máquinas lo que los idiomas, los gestos, o el lenguaje corporal son a la comunicación entre humanos. Así, de igual manera que dos humanos necesitan hablar el mismo idioma para poder entenderse, los dispositivos necesitan utilizar los mismos protocolos para IoT para intercambiar información.

Para su funcionamiento, los protocolos de datos emergentes utilizados en las redes de IoT cuentan con varios niveles:

- *Aplicación: la interfaz entre usuario y dispositivo.*
- *Red: potencia la comunicación entre enrutador y cada uno de los dispositivos conectados a la red.*
- *Transporte: facilita la comunicación de datos entre los diferentes niveles y garantiza su seguridad.*
- *Físico: la red de comunicación física entre dispositivos.*
- *Vínculo de datos: encargado de transportar los datos en el sistema y detectar y corregir problemas*

En la comunicación de dispositivos IoT con internet, los protocolos más habituales son MQTT, CoAP y HTTP. Son altamente flexibles ya que están pensados para transmitir cualquier tipo de información.

Además de los conocidos protocolos HTTP, aquí destacan:

- *MQTT (MQ Telemetry Transport). Sigue un modelo de publicación-suscripción, permitiendo la comunicación entre un gran número de dispositivos. Para su*

funcionamiento, un servidor central llamado broker se encarga de recibir los mensajes de los dispositivos emisores y distribuirlos entre los receptores. Los mensajes, además, se organizan por etiquetas de forma jerárquica.

- *CoAP (Constrained Application Protocol) se orienta a la comunicación entre dispositivos de baja potencia y emplea el modelo REST de HTTP, junto a otros requisitos como multicast, soporte UDP y bajo overhead." (barbaraiot, 2021)*

En este reto ocuparemos el protocolo de comunicación MQTT, este protocolo va a ser el encargado de mandar la información obtenida por el sensor del ESP32 a la plataforma ThingSpeak.

Propuesta:

La siguiente propuesta implementa una solución ocupando el concepto de IoT (Internet of Things) para atacar una problemática. La problemática es la logística que se tiene en las cadenas de suministros y la solución que proponemos es convertirlas a redes logísticas inteligentes, esto quiere decir que a lo largo de la cadena de suministro se pueda monitorear en tiempo real las variables y condiciones del producto a exportar y también analizar sus datos posteriormente apoyándonos en la arquitectura de nube.

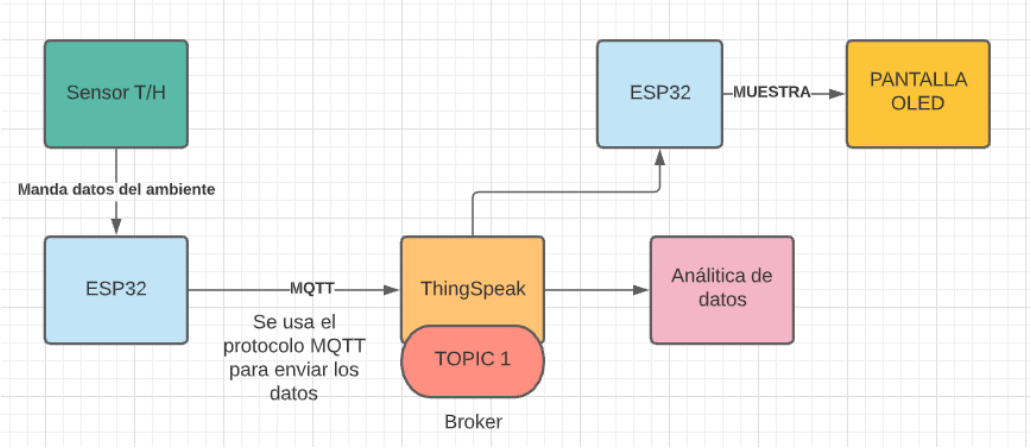
El enfoque va principalmente a las redes logísticas que se enfocan en exportar productos perecederos, poniendo como ejemplo específico el aguacate. Una de las variables más importantes de este producto a monitorear son la humedad y temperatura. Con esto también se pretende apoyar a los ODS mediante la implementación de tecnología, reducir costos de monitoreo y análisis y hacer esto accesible para países en desarrollo.

Los materiales y plataformas que vamos a necesitar van a ser los siguientes:

- ESP32
- Sensor de humedad y temperatura DHT11
- Pantalla Oled Ssd1306 de 128x64
- Jumpers
- Protoboard
- Led

- Resistencias
- Cable Micro USB
- Plataforma Arduino IDE
- Plataforma de nube ThingSpeak

A continuación se presenta un diagrama de como sería la implementación de la solución mediante un diagrama de bloques:



Es importante mencionar que en este proyecto utilizaremos el protocolo de comunicación MQTT. Y para conectarnos con la pantalla oled se va a utilizar el protocolo de comunicación I2C. Todo esto viene en el código que se anexa en el apéndice al final de este documento.

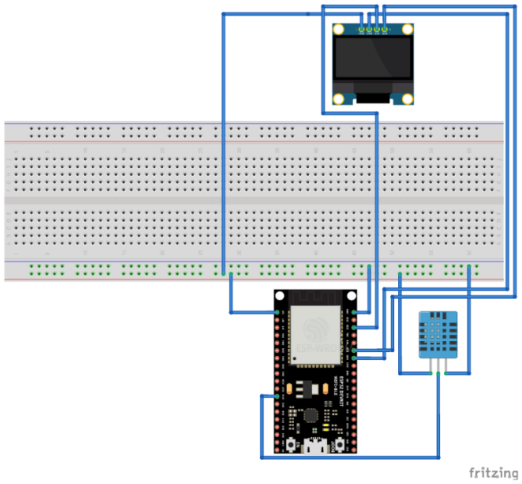
El primer paso fue crear una cuenta en ThingSpeak, para esto debimos crear un canal y guardar las contraseñas y número de usuario que se genera, para posteriormente poder ingresar esto en el código y tener permiso del broker para publicar las variables de temperatura y humedad.

Es necesario habilitar en el canal de ThingSpeak dos campos para los propósitos de este proyecto, el primer campo se va a llamar humedad y el otro recibirá el nombre de temperatura.

Después de esto descargamos el controlador CP102X para conectarnos de manera serial con el ESP32, ya que sin este controlador no se puede enviar datos de manera simple con tan solo el USB.

También es necesario descargar las librerías correspondientes al sensor DHT11 y de la pantalla OLED, las librerías que se descargan van a ser de la empresa de Software Adafruit.

Para la conexión se utilizó el siguiente diagrama de conexión:



Y para la conexión del led se conectó una resistencia al negativo y el lado positivo al pin 19 del ESP32.

Las funciones del código se explican a continuación (este código se encuentra en el apéndice al final del documento) :

-Función que establece la conexión.

-Nos suscribimos para recibir la información del broker.

-Función que lee en segundo plano la información que se genere en broker. Por si hay algún cambio en el tópico.

-Mandamos la información que se quiere publicar.

-Aplicamos un condicional y si la temperatura es mayor a 20 grados celsius se prende un led color rojo de alerta.

-Aplicamos un condicional y si la temperatura es menor a 20 grados celsius se mantiene apagado o se apaga el led de alerta.

-Mandamos la información a la pantalla OLED y ajustamos el tamaño de los píxeles.

También se aplicó código de Matlab, después de que el broker recibía los datos, se analizaron los datos de temperatura y humedad para poder hacer dos tipos de gráficas, una gráfica que muestra la temperatura en los últimos 3 días y un histograma con los registros de la temperatura. (Estos códigos se encuentran indexados en el apéndice al final del documento).

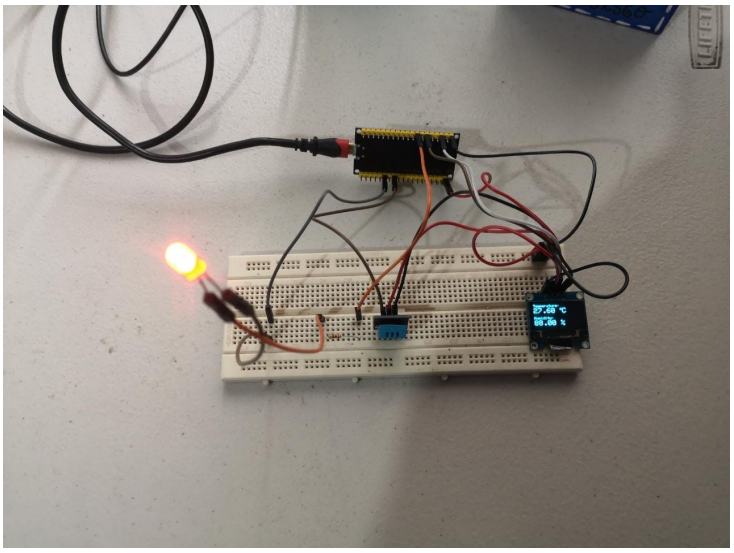
La temperatura óptima para la conservación de los aguacates es de 20 grados centígrados, después de esta temperatura los aguacates comienzan un proceso de descomposición más acelerado.

Lo que hace este dispositivo es sensar la temperatura y humedad de los aguacates y en el momento que se detecte una temperatura mayor a esta se activa una alarma física y otra en la plataforma web. La alerta física es el led color rojo que se enciende en forma de alerta y la alerta en la web es un widget de foco rojo que se activa cuando se ha sobrepasado este límite.

De igual manera se programó para cuando se pase dicha temperatura se publique en Twitter que se ha sobrepasado el límite y cuando la temperatura haya sido restablecida se publica que se ha corregido.

Además la plataforma te permite analizar los datos que se obtuvieron conforme el paso del tiempo para sacar conclusiones en cuestión de calidad del producto y qué acciones se deberían tomar.

A continuación se muestra una foto del proyecto, con el foco led encendido debido a una temperatura excesiva:



Resultados experimentales:

Para correr el experimento lo que se hizo fue conectar la ESP32 al wifi, se corroboró que estuviera midiendo la temperatura y humedad adecuadamente y enviando los datos al broker de la manera correspondiente.

Se dejó trabajando al sensor durante 20 minutos, bajo un ambiente controlado, esto porque alteraba la temperatura y humedad a conveniencia exhalando cerca del sensor para monitorear su comportamiento.

Se puede observar que la temperatura que más se registró durante ese lapso de tiempo es de 15°C, siendo la máxima temperatura de hasta 30°C. La temperatura vemos que decrece de manera exponencial al llegar a su límite, como lo estipula Newton en la ley del enfriamiento.

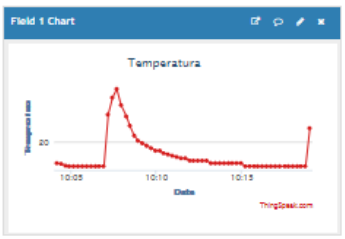
La humedad la mayoría del tiempo se mantuvo constante y se puede observar que realmente no existe una correlación entre la temperatura y la humedad, ya que se midieron diferentes temperaturas con el mismo porcentaje de humedad entre todas ellas.

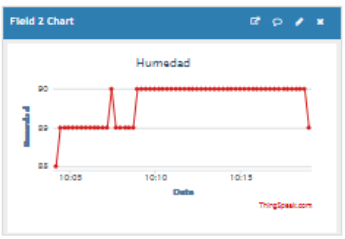
Esto mismo se puede observar en la gráfica temperatura-humedad (eje Y), tiempo (eje X), cuando la temperatura aumenta en el tiempo la humedad no sigue un patrón de crecimiento o decrecimiento consistente.

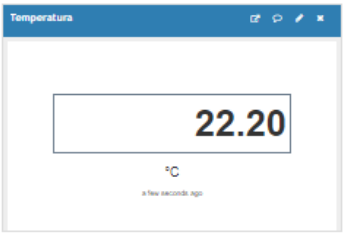
Todo esto se puede observar en las gráficas en la parte de abajo. Estas fueron hechas mediante código en MATLAB.

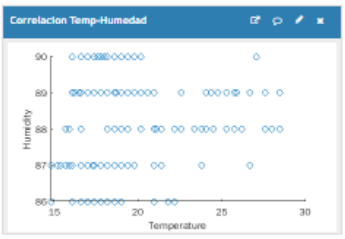
También se experimentó con los tweets, al forzar un incremento de temperatura superior a 20°C se publicaba un tweet advirtiendo el alza de temperatura y cuando se dejaba enfriar a temperatura ambiente se publicaba otro tweet notificando que la temperatura se había regulado nuevamente.

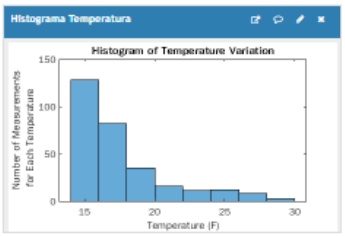
También experimentamos con diferentes tiempos de delay en el código, estrictamente en la variable Update Interval, ya que el sensor requiere de cierto tiempo de tolerancia para poder hacer una medición confiable. Finalmente optamos por 10 segundos, ya que de acuerdo a las especificaciones técnicas del sensor, la medición mínima es de 6 segundos, la típica de 10 y la máxima de 15. Por eso decidimos quedarnos con una medición rápida, pero confiable.

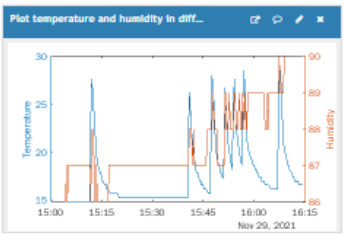












Twitter Account	API Key	Action
a01730560	HD239E8NTGTOEORF	Regenerate API Key Unlink Account

Reacts using ThingTweet

Name	Message	Last Sent	Twitter Account
Temperatura extrema	Temperatura excesiva, tomar acciones	2021-11-29 15:57	a01730560
Temperatura corregida	La temperatura ha sido corregida	2021-11-29 15:56	a01730560

Conclusiones:

Creemos que esta implementación de las IoT en este tipo de empresas beneficiará a demasiadas empresas de este tipo, ya que al innovar y al poder tener esa adaptabilidad con este nuevo tipo de tecnologías, podrán generar más ingresos, mayor inmersión al mercado, etc.
Pero esto hablando de lo económico, ahora hablando de la tecnología, con las IoT se empiezan a innovar para poder así no quedar atrás en el mercado, ofreciendo a sus clientes una mayor confianza de que se están implementando estas nuevas tecnologías.

En cuestión del reto, esta implementación fue una excelente idea, ya que se nos facilitó la forma de poder juntar la incógnita del reto con una necesidad de la vida real que se necesita con la implementación del IoT y nos sentimos satisfechos de poder solucionar la problemática de la situación problema y poder dar un vistazo de lo que puede ser de este tipo de empresas con la implementación del IoT y así poder innovar y no quedar fuera de sus competidores, además sabiendo que este mercado es muy concurrido y lo que creemos es que el que pueda implementar la IoT primero antes que sus competidores va a ser el que avanza más en el mercado, claros ejemplos como; DHL, Amazon, etc.

Nuestros logros principales fueron aprender a ocupar Arduino IDE, programar la ESP32, interpretar y analizar datasheets de sensores y componentes electrónicos, instalar librerías y drivers necesarios para la lectura y escritura. De igual manera lograr una conexión mediante el protocolo MQTT que fue implementado del ESP32 al broker. También algo muy importante que logramos fue la interpretación y análisis de datos mediante MATLAB.

Algunas limitaciones del proyecto son la frecuencia con la que se actualiza debido a la versión de prueba de ThingSpeak, que el sensor tiene un margen de error porque no es uno muy especializado, es uno barato, que estamos ocupando la conexión wifi y esto limita la movilidad del dispositivo y el tamaño con el que cuenta, ya que sería mejor tener el circuito en una placa impresa para no ocupar los jumpers y la protoboard.

Referencias:

¿Qué es IoT (Internet Of Things)?. (2021). Retrieved 26 November 2021, from https://www2.deloitte.com/es/es/pages/technology/articles/IoT-internet-of-things.html

¿Qué es un Sensor y Qué Hace? | Dewesoft. (2021). Retrieved 26 November 2021, from https://dewesoft.com/es/daq/que-es-un-sensor

IoT Cloud Architecture Insights – Why Database Design Matters. (2021). Retrieved 26 November 2021, from https://www.embitel.com/iot-insights/iot-cloud-architecture-insights-why-database-design-matters

Lee, I., & Lee, K. (2015). The Internet of Things (IoT): Applications, investments, and challenges for enterprises. *Business Horizons*, *58*(4), 431-440

Objetivos y metas de desarrollo sostenible. (2021). Retrieved 26 November 2021, from https://www.un.org/sustainabledevelopment/es/objetivos-de-desarrollo-sostenible/

Protocolos de comunicación en IoT que deberías conocer. (2021). Retrieved 26 November 2021, from https://barbaraiot.com/blog/protocolos-iot-que-deberias-conocerr/

Ramírez, L. G. C., Jiménez, G. S. A., & Carreño, J. M. (2014). *Sensores y actuadores*. Grupo Editorial Patria.

Sethi, P., & Sarangi, S. R. (2017). Internet of things: architectures, protocols, and applications. *Journal of Electrical and Computer Engineering*, *2017*.

Apéndice:

Código en Arduino IDE:

```
//Project IoT
//Francisco Rocha Juárez. A01730560
//Horacio Iann Toquiantzi Escarcega. A01734839
//Código tomado de clase con algunas implementaciones.
//Este código no es creación 100% nuestra.
//Código por el profesor Cesar Torres

#include <Wire.h>                 // Library required for esp32 S2C communications
#include <DHT.h>                  // Library required for DHT11 sensor
#include <Adafruit_GFX.h>         // Library required for the OLED display
#include <Adafruit_SSD1306.h>
#include <Adafruit_Sensor.h>

#include <PubSubClient.h>         // Library required for MQTT

//We establish ledpin 19 to turn the red light
const int LEDPIN = 19;

// Set up the particular type of sensor used (DHT11, DHT21, DHT22, ...) attached to the
esp32 board.
#define DHTPIN 14             // Digital esp32 pin to receive data from DHT.
#define DHTTYPE    DHT11      // DHT 11 is used.
DHT dht (DHTPIN, DHTTYPE);    // DHT sensor setup.

// Variables to hold temperature (t) and humidity (h) readings
float t;
float h;

// Set up an SSD1306 display connected to I2C (SDA -->, SCL -->)
#define SCREEN_WIDTH 128          // OLED display width  (pixels)
#define SCREEN_HEIGHT 64          // OLED display height (pixels)
```

```
Adafruit_SSD1306 display (SCREEN_WIDTH, SCREEN_HEIGHT, &Wire, -1);



// Credentials to connect to your WiFi Network.
// Strongly related to your working envirnonment
char ssid[] = "WS5200";
char pass[] = "*Misifus07/09";

// MQTT broker server. Note that the specific port is defined depending on the type of the
// used connection.
const char* server = "mqtt3.thingspeak.com";

// A macro is used to select between secure and nonsecure connection as it is
hardware-dependent
// Certain IoT hardware platforms do not work with the WiFiClientSecure library.

#define USESECUREMQTT            // Comment this line if non secure connection is used

#ifdef USESECUREMQTT
  #include <WiFiClientSecure.h>
  #define mqttPort 8883
  WiFiClientSecure client;
#else
  #include <WiFi.h>
  #define mqttPort 1883
  WiFiClient client;
#endif

// Credentials that allow to publish and subscribe to the ThingSpeak channel. It depends on
your
// ThinkSpeak account and the defined channels
const char mqttUserName[]   = "IA8TJi4TCy4rHDMWKDQbEBw";
const char clientID[]       = "IA8TJi4TCy4rHDMWKDQbEBw";
const char mqttPass[]       = "egUH11vG40ls/QecuPPIfoIA";
```

// Channel ID that is defined in the ThinkSpeak account to hold out streaming data. Recall that
// up to eight fields are allowed per channel
#define channelID 1587240 // This channel holds two fields: temperature and humidity

// Since the target esp32-based board supports secure connections, a thingspeak certificate is used.

```
const char * PROGMEM thingspeak_ca_cert = \
"-----BEGIN CERTIFICATE-----\n" \

"MIIDxTCCAq2gAwIBAgIQAqxcJmoLQJuPC3nyrkYldzANBgkqhkiG9w0BAQUFADBs\n" \

"MQswCQYDVQQGEwJVUzEVMBMGA1UEChMMRGlnaUNlcnQgSW5jMRkwFwYDVQQLExB3\n" \

"d3cuZGlnaWNlcnQuY29tMSswKQYDVQQDEyJEaWdpQ2VydCBIaWdoIEFzc3VyYW5j\n" \

"ZSBFViBSb290IENBMB4XDTA2MTExMDAwMDAwMFoXDTMxMTExMDAwMDAwMFowbDEL\n" \

"MAkGA1UEBhMCVVMxFTATBgNVBAoTDERpZ2lDZXJ0IEluYzEZMBcGA1UECxMQd3d3\n" \

"LmRpZ2ljZXJ0LmNvbTErMCkGA1UEAxMiRGlnaUNlcnQgSGlnaCBBc3N1cmFuY2Ug\n" \

"RVYgUm9vdCBDQTCCASIwDQYJKoZIhvcNAQEBBQADggEPADCCAQoCggEBAMbM5XPm\n" \
```

```
  "+9S75S0tMqbf5YE/yc0lSbZxKsPVlDRnogocsF9ppkCxxLeyj9CYpKlBWTrT3JTW\n" \

"PNt0OKRKzE0lgvdKpVMSOO7zSW1xkX5jtqumX8OkhPhPYlG++MXs2ziS4wblCJEM\n
" \

"xChBVfvLWokVfnHoNb9Ncgk9vjo4UFt3MRuNs8ckRZqnrG0AFFoEt7oT61EKmEFB\n"
\

"Ik5lYYeBQVCmeVyJ3hlKV9Uu5l0cUyx+mM0aBhakaHPQNAQTXKFx01p8VdteZOE3\n
" \
  "hzBWBOURtCmAEvF5OYiiAhF8J2a3iLd48soKqDirCmTCv2ZdlYTBoSUeh10aUAsg\n"
\

"EsxBu24LUTi4S8sCAwEAAaNjMGEwDgYDVR0PAQH/BAQDAgGGMA8GA1UdEwEB
/wQF\n" \

"MAMBAf8wHQYDVR0OBBYEFLE+w2kD+L9HAdSYJhoIAu9jZCvDMB8GA1UdIwQ
YMBaA\n" \

"FLE+w2kD+L9HAdSYJhoIAu9jZCvDMA0GCSqGSIb3DQEBBQUAA4IBAQAcGgaX3N
ec\n" \

"nzyIZgYIVyHbIUf4KmeqvxgydkAQV8GK83rZEWWONfqe/EW1ntlMMUu4kehDLI6z\n"
\

"eM7b41N5cdblIZQB2lWHmiRk9opmzN6cN82oNLFpmyPInngiK3BD41VHMWEZ71jF\n
" \

"hS9OMPagMRYjyOfiZRYzy78aG6A9+MpeizGLYAiJLQwGXFK3xPkKmNEVX58Svnw2
\n" \

"Yzi9RKR/5CYrCsSXaQ3pjOLAEFe4yHYSkVXySGnYvCoCWw9E1CAx2/S6cCZdkGCe\
n" \
  "vEsXCS+0yx5DaMkHJ8HSXPfqIbloEpw8nL+e/IBcm2PN7EeqJSdnoDfzAIJ9VNep\n" \
```

```
  "+OkuE6N36B9K\n" \
  "-----END CERTIFICATE-----\n";


// Initial state of the wifi connection
int status = WL_IDLE_STATUS;

// The MQTT client is liked to the wifi connection
PubSubClient mqttClient( client );

// Variables are defined to control the timing of connections and to define the
// update rate of sensor readings (in milliseconds)

int connectionDelay    = 4;    // Delay (s) between trials to connect to WiFi
long lastPublishMillis = 0;    // To hold the value of last call of the millis() function
int updateInterval     = 15;   // Sensor readings are published every 15 seconds o so.


/*****************************************************************************
*****************************
* Function prototypes mainly grouped by functionality and dependency
*****************************************************************************
*****************************/

// Function to initialize the SSD1306 OLED display if available
int initDisplay();

// Function to display local temperature and humidity on the OLED display
void showInDisplay(float t, float h);

// Function to connect to WiFi.
void connectWifi();

// Function to connect to MQTT server, i.e., mqtt3.thingspeak.com
```

```
void mqttConnect();

// Function to subscribe to ThingSpeak channel for updates.
void mqttSubscribe( long subChannelID );

// Function to handle messages from MQTT subscription to the ThingSpeak broker.
void mqttSubscriptionCallback( char* topic, byte* payload, unsigned int length );

// Function to publish messages to a ThingSpeak channel.
void mqttPublish(long pubChannelID, String message);



void setup() {
  // Recall that this code is executed only once.
  // Establish the serial transmission rate and set up the communication
  Serial.begin( 115200 );
  // Some delay to allow serial set up.
  delay(3000);

  dht.begin();                // Initialize DHT11 sensor
  initDisplay();              // Initialize the OLED display

  pinMode(LEDPIN,OUTPUT);       //LEDPIN is going to be output



  // Connect to the specified Wi-Fi network.
  connectWifi();

  // Configure the MQTT client to connect with ThinkSpeak broker
  mqttClient.setServer( server, mqttPort );

  // Set the MQTT message handler function.
  mqttClient.setCallback( mqttSubscriptionCallback );
  // Set the buffer to handle the returned JSON.
```

```
  // NOTE: A buffer overflow of the message buffer will result in your callback not being
invoked.
  mqttClient.setBufferSize( 2048 );

  // Use secure MQTT connections if defined.
  #ifdef USESECUREMQTT
    // Handle functionality differences of WiFiClientSecure library for different boards.
    // Herein we are dealing with esp32-based IoT boards.
      client.setCACert(thingspeak_ca_cert);
  #endif
}

void loop() {
  // After everything is set up, go the perception-action loop
  // Reconnect to WiFi if it gets disconnected.
  if (WiFi.status() != WL_CONNECTED) {
      connectWifi();
  }

  // Connect if the MQTT client is not connected and resubscribe to channel updates.
  // ThinkSpeak broker server : subscribe to the specified channel
  if (!mqttClient.connected()) {
    mqttConnect();
    mqttSubscribe( channelID );
  }

  // Call the loop to maintain connection to the server.
  mqttClient.loop();

  // Update the ThingSpeak channel periodically according to the specified rate.
  // The update results in the message to the subscriber.
  if ( abs(millis() - lastPublishMillis) > updateInterval*1000) {
    // Sensors readings: temperature and humidity
    t = dht.readTemperature();
```

```
    h = dht.readHumidity();
    if ( isnan(t) || isnan(h)) {
        Serial.println("Failed to read from DHT sensor!");
      }
      Serial.print(F("Local temperature: "));
      Serial.print(t);
      Serial.print(" ºC ");
      Serial.print(F("  Local humidity: "));
      Serial.print(h);
      Serial.println(" %");
    //If the temperature is greater than 20° then turn on the red led
    if (t>20){
      digitalWrite(LEDPIN,HIGH);
    } else {
      digitalWrite(LEDPIN,LOW);
    }
    // Show the sensor readings in the OLED local display
    showInDisplay(t,h);
    //mqttPublish( channelID, (String("field1=")+String(WiFi.RSSI())) );
    mqttPublish( channelID, (String("field1=")+String(t) + String("&field2=")+String(h) ) );
    lastPublishMillis = millis();
  }
}


// Function to initialize the SSD1306 OLED display
int initDisplay()
{
  if (!display.begin(SSD1306_SWITCHCAPVCC, 0x3C)) {
    Serial.println(F("SSD1306 allocation failed"));
    for(;;);
  }
  delay(2000);
  display.clearDisplay();
```

```
  display.setTextColor(WHITE);
}


// Function to display local temperature and humidity on the SSD1306 OLED display
void showInDisplay(float t, float h)
{
  // Clear display
  display.clearDisplay();
  // display temperature
  display.setTextSize(1);
  display.setCursor(0,0);
  display.print("Temperature: ");
  display.setTextSize(2);
  display.setCursor(0,10);
  display.print(t);
  display.print(" ");
  display.setTextSize(1);
  display.cp437(true);
  display.write(167);
  display.setTextSize(2);
  display.print("C");

  // display humidity
  display.setTextSize(1);
  display.setCursor(0, 35);
  display.print("Humidity: ");
  display.setTextSize(2);
  display.setCursor(0, 45);
  display.print(h);
  display.print(" %");
  display.display();
}
```

```
// Function to connect to WiFi.
void connectWifi()
{
  Serial.println( "Connecting to Wi-Fi..." );
  // Loop until WiFi connection is successful
    while ( WiFi.status() != WL_CONNECTED ) {
    WiFi.begin( ssid, pass );
    delay( connectionDelay*1000 );
    Serial.println( WiFi.status() );
  }
  Serial.println( "Connected to Wi-Fi." );
}

// Function to connect to MQTT server.
void mqttConnect() {
  // Loop until the client is connected to the server.
  while ( !mqttClient.connected() )
  {
    // Connect to the MQTT broker.
    if ( mqttClient.connect( clientID, mqttUserName, mqttPass ) ) {
      Serial.print( "MQTT to " );
      Serial.print( server );
      Serial.print (" at port ");
      Serial.print( mqttPort );
      Serial.println( " successful." );
    } else {
      Serial.print( "MQTT connection failed, rc = " );
      // See https://pubsubclient.knolleary.net/api.html#state for the failure code explanation.
      Serial.print( mqttClient.state() );
      Serial.println( " Will try the connection again in a few seconds" );
      delay( connectionDelay*1000 );
    }
  }
}
```

```
// Function to subscribe to ThingSpeak channel for updates.
void mqttSubscribe( long subChannelID ){
  String myTopic = "channels/"+String( subChannelID )+"/subscribe";
  mqttClient.subscribe(myTopic.c_str());
}

// Function to handle messages from MQTT subscription to the ThingSpeak broker.
void mqttSubscriptionCallback( char* topic, byte* payload, unsigned int length ) {
  // Print the message details that was received to the serial monitor.
  Serial.print("Message arrived from ThinksSpeak broker [");
  Serial.print(topic);
  Serial.print("] ");
  for (int i = 0; i < length; i++) {
    Serial.print((char)payload[i]);
  }
  Serial.println();
}

// Function to publish messages to a ThingSpeak channel.
void mqttPublish(long pubChannelID, String message) {
  String topicString ="channels/" + String( pubChannelID ) + "/publish";
  mqttClient.publish( topicString.c_str(), message.c_str() );
}
```

Código para histograma (Matlab):

```
% Read temperature for the last 10 hours from a ThingSpeak channel
and
% visualize temperature variations using the MATLAB HISTOGRAM
function.

readChannelID = 1587240;

% Temperature Field ID
TemperatureFieldID = 1;
```

```
% Channel Read API Key
% If your channel is private, then enter the read API
% Key between the '' below:
readAPIKey = 'X2I2153G69SI89CS';

% Get temperature data from field 4 for the last 10 hours = 10 x 60
% minutes. Learn more about the THINGSPEAKREAD function by going to
% the Documentation tab on the right side pane of this page.

tempF = thingSpeakRead(readChannelID,'Fields',TemperatureFieldID,...
'NumMinutes',10*60, 'ReadKey',readAPIKey);

histogram(tempF);
xlabel('Temperature (F)');
ylabel('Number of Measurements\newline for Each Temperature');
title('Histogram of Temperature Variation');
```

Código correlación Temperatura-Humedad (MATLAB):

```
% Read temperature and humidity from a ThingSpeak channel and
visualize the
% relationship between them using the SCATTER plot

% Channel ID to read data from
readChannelID = 1587240;
% Temperature Field ID
TemperatureFieldID = 1;
% Humidity Field ID
HumidityFieldID = 2;

% Channel Read API Key
% If your channel is private, then enter the read API
% Key between the '' below:
readAPIKey = 'X2I2153G69SI89CS';

% Read Temperature and Humidity Data. Learn more about the
THINGSPEAKREAD function by
% going to the Documentation tab on the right side pane of this page.
```

```
data = thingSpeakRead(readChannelID,'Fields',[TemperatureFieldID
HumidityFieldID], ...
                                                'NumPoints',300, ...
                                                'ReadKey',readAPIKey);

temperatureData = data(:,1);

% Read Humidity Data
humidityData = data(:,2);

% Visualize the data
scatter(temperatureData,humidityData);
xlabel('Temperature');
ylabel('Humidity');
```

Código graficar variables en ejes distintos (MATLAB):

```
% Read temperature and humidity data from a ThingSpeak channel and
% visualize the data on the same plot with different Y-Axes using the
% YYAXIS and PLOT functions


% Channel ID to read data from
readChannelID = 1587240;
% Temperature Field ID
TemperatureFieldID = 1;
% Wind speed Field ID
humidityFieldID = 2;

% Channel Read API Key
% If your channel is private, then enter the read API
% Key between the '' below:
readAPIKey = 'X2I2153G69SI89CS';

% Read Data. Learn more about the thingSpeakRead function by
% going to the Documentation tab on the right side pane of this page.
[data, timeStamps] = thingSpeakRead(readChannelID,
'Fields',[TemperatureFieldID humidityFieldID], ...

'NumPoints', 300, ...
```

```
                                                    'ReadKey',
readAPIKey);

% Extract the temperature data from the first column
temperatureData = data(:, 1);
% Extract the humidity data from the second column
humidityData = data(:, 2);

% Visualize Data
yyaxis left
plot(timeStamps, temperatureData);
ylabel('Temperature');

yyaxis right
plot(timeStamps, humidityData);
ylabel('Humidity');
```

Contribuciones del Equipo:

Francisco Rocha Juárez: Armado del dispositivo, Código Matlab Correlación, Código Matlab Temperatura-Humedad, realizar pruebas, sincronizar ThingSpeak con Twitter, realizar conexión mediante el protocolo MQTT.

Horacio Iann Toquiantzi Escárcega: Programación del encendido del led rojo, Código Matlab histograma, agregar widgets a ThingSpeak, código contraseñas API, investigar como hacer la conexión wifi, widget humedad en ThingSpeak.

Links vídeos individuales (Carpeta Drive) :

https://drive.google.com/drive/folders/1S7dUIXj4IQxxZ5jSh5NLGYXJA1vR6LL8?usp=sharing