

UNIVERSIDAD NACIONAL MAYOR DE SAN MARCOS

(Universidad del Perú, DECANA DE AMÉRICA)

Facultad de Ingeniería de Sistemas e Informática

E. P. Ingeniería de Sistemas



Tarea de Laboratorio 2

DOCENTE: Guerra Guerra Jorge Leoncio

CURSO: Internet de las cosas

SECCIÓN: 1

GRUPO: 1

INTEGRANTES:

Alfaro Mauricio, Kevin Johan	20200242
Arrieta Palacios, Andrés Leonardo	20200153
Taipe Javier, Luis Angel	20200118

Lima, Perú

2023

Índice

1. Introducción	3
2. Diseño y metodología	4
2.1. Configuración del hardware	4
2.2. Configuración del software	5
3. Resultados	17
4. Conclusiones	21
5. Referencias	22

1. Introducción

La creciente adopción de dispositivos IoT (Internet de las cosas) y la necesidad de soluciones de alerta eficaces han llevado al desarrollo de proyectos innovadores que combinan hardware, software y servicios en línea. Este paper presenta un proyecto que aborda la necesidad de conectar el microcontrolador ESP32 a la nube de Google (Google Cloud Platform). Para ello utilizaremos datos en streaming (tiempo real) generados por el sensor DHT11, tanto de temperatura como de humedad ambiental.

La necesidad de notificaciones en tiempo real en situaciones críticas, como la detección de sonidos o movimientos inesperados, ha impulsado la búsqueda de soluciones que aprovechen la popularidad y accesibilidad de las aplicaciones de mensajería instantánea.

El objetivo principal de este proyecto es proporcionar una solución eficiente y asequible para la generación de data utilizando un ESP32, un sensor de temperatura y humedad junto con otros componentes clave. Esto puede ser de gran utilidad en una variedad de aplicaciones, como el constante seguimiento del clima.

El presente trabajo abordará la implementación detallada de esta solución, destacando la configuración del hardware y el software, la conexión con la nube, y sus detalles. Además, se discutirán los resultados obtenidos y las posibles aplicaciones y limitaciones de este sistema.

2. Diseño y metodología

En esta sección se proporciona una descripción del diseño y la metodología implementados en el proyecto, centrándose en los componentes específicos utilizados y su interconexión.

2.1. Configuración del hardware

El diseño del hardware es un elemento crítico para el funcionamiento del proyecto. Se han empleado los siguientes componentes:

- **Arduino ESP32:** Se utilizó la placa de desarrollo ESP32 de 38 Pines que integra el microcontrolador ESP32-WROOM-32 SMD de Espressif. Esta placa permite controlar todo tipo de sensores, módulos y actuadores mediante WIFI y BLUETOOTH, para proyectos de Internet de las cosas “IoT” de forma eficiente y económica.

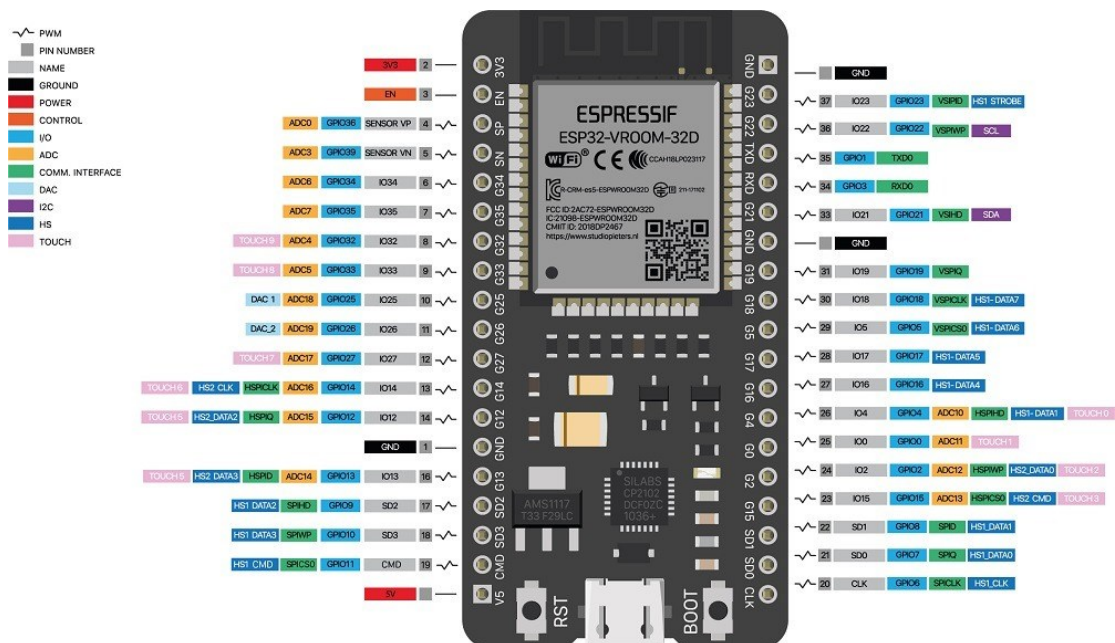


Figura N.º 1. Vista detallada de ESP-32

- **Sensor DH11:** Se utilizó el sensor de humedad y temperatura DH11 el cual utiliza un sensor capacitivo de humedad y un termistor para medir el aire circundante, y muestra los datos mediante una señal digital en el pin de datos (no posee salida analógica). Se conecta el cable positivo (+) al pin de 5V del ESP32, el negativo (-) al GND del ESP32 y el de *dh13* al pin IO13 de la placa.

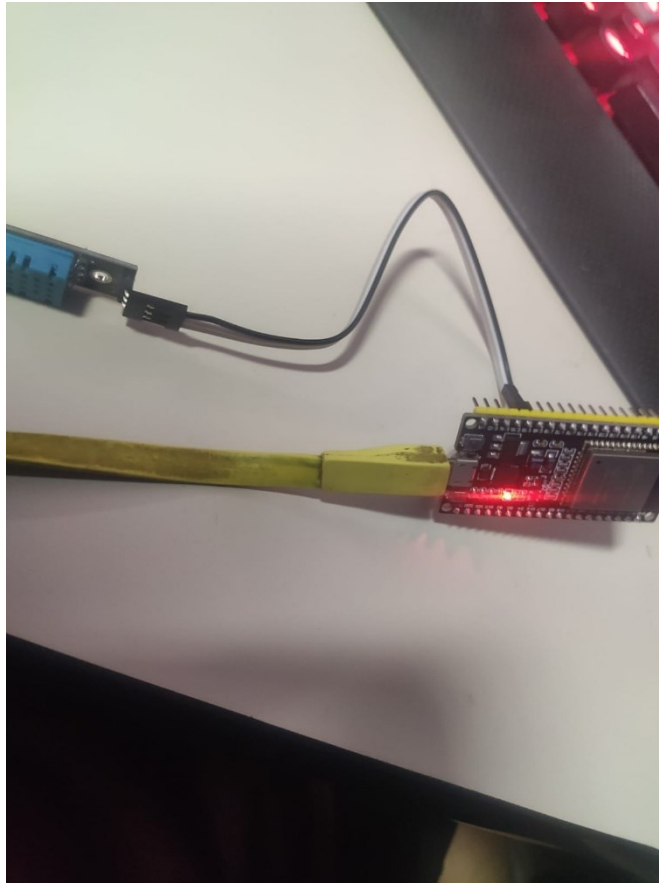


Figura N.º 2. Vista de sensor DH11 conectado a ESP32

2.2. Configuración del software

El software del proyecto se basa en el entorno de desarrollo Arduino. A continuación, se describen los aspectos clave de la configuración del software:

- **Librerías ,definiciones e inicialización del sensor:** Se incluye *thingProperties.h* que contiene propiedades y configuraciones específicas del proyecto o del dispositivo en uso; también *<DHT.h>* que se utiliza para trabajar con sensores de temperatura y humedad DHT. Luego tenemos la definición de las constantes *DHTPIN* como 13 y *DHTTYPE* como DHT11. Finalmente se crea una instancia de la clase *DHT* llamada "dht" utilizando el pin definido *DHTPIN* y el tipo de sensor definido *DHTTYPE* (DHT11 en este caso). Esta instancia se utilizará para interactuar con el sensor DHT en el programa.

```
#include "thingProperties.h"

#include <DHT.h>

#define DHTPIN 13 // Pin al que está conectado el sensor DHT11

#define DHTTYPE DHT11

DHT dht(DHTPIN, DHTTYPE);
```

- **Inicialización del setup():** Empieza inicializando la comunicación serie con una velocidad de 115200 baudios para la depuración a través del monitor serie, luego agrega un retraso de 5 segundos para permitir la conexión con el monitor serie, lo que facilita la depuración. Inicializa el sensor DHT para medir la temperatura y la humedad para luego llamar a la función *initProperties()* que se encarga de inicializar las propiedades relacionadas con la comunicación con la plataforma Arduino IoT Cloud. En la siguiente línea inicia la conexión con Arduino IoT Cloud utilizando la preferencia de conexión especificada (configurada previamente en el archivo *thingProperties.h*) para seguir estableciendo el nivel de mensajes de depuración en 2, lo que proporciona información detallada sobre la conexión con la nube y los errores. Finalmente con *ArduinoCloud.printDebugInfo()* imprime información de depuración en el monitor serie relacionada con la conexión y el estado de la nube de Arduino IoT Cloud.

```

void setup() {

    // Initialize serial and wait for port to open:

    Serial.begin(115200);

    // This delay gives the chance to wait for a Serial Monitor without
    blocking if none is found

    delay(5000);

    dht.begin();

    // Defined in thingProperties.h

    initProperties();

    // Connect to Arduino IoT Cloud

    ArduinoCloud.begin(ArduinoIoTPreferredConnection);

    /*

    The following function allows you to obtain more information
    related to the state of network and IoT Cloud connection and errors
    the higher number the more granular information you'll get.
    The default is 0 (only errors).
    Maximum is 4

    */

    setDebugMessageLevel(2);

    ArduinoCloud.printDebugInfo();

}

```

- Inicialización del loop():** Empieza inicializando el *ArduinoCloud.update()* que mantiene la comunicación con la plataforma Arduino IoT Cloud (debe llamarse de manera regular en el bucle para permitir la actualización de los datos y la interacción con la nube). Luego lee la humedad y la temperatura del sensor DHT y almacena los valores en las variables *h* y *t*. Sigue con la impresión de la humedad y la temperatura en el monitor serie para propósitos de depuración para luego asignar los valores de humedad y temperatura a las variables que se utilizan para enviar estos datos a la plataforma Arduino IoT Cloud. Esto permite que los valores medidos se envíen a la nube para su posterior visualización o análisis.

```

void loop() {

  ArduinoCloud.update();

  // Your code here

  float h = dht.readHumidity();

  float t = dht.readTemperature();

  Serial.print("Humedad: ");

  Serial.print(h);

  Serial.print(" %\t Temperatura: ");

  Serial.print(t);

  Serial.println(" °C");

  humidity = h;

  temperature = t;

  message="Temperature= "+ String(temperature)+" Humidity = " +
String(humidity);

}

```

- **Configuración general del Arduino IOT Cloud**
 - **Inicio de Sesión:** Se inicia sesión con cuenta de Arduino ó de otras plataformas. Para este trabajo se utilizó el correo institucional.

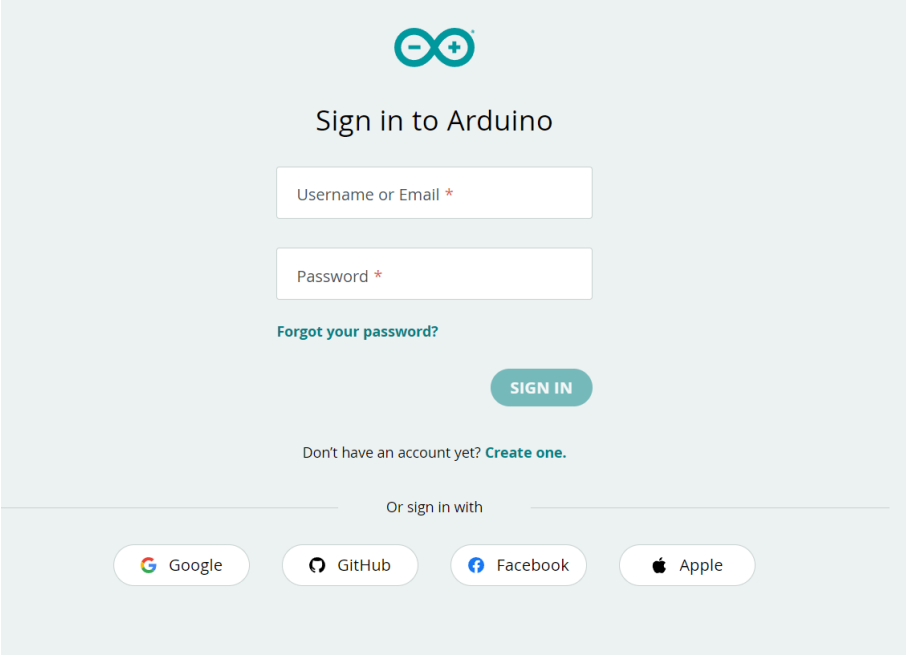


Figura N.º 3. Inicio de sesión Arduino

- **Creación de Things:** Pantalla inicial donde se crearán las variables principales y se dará la configuración.

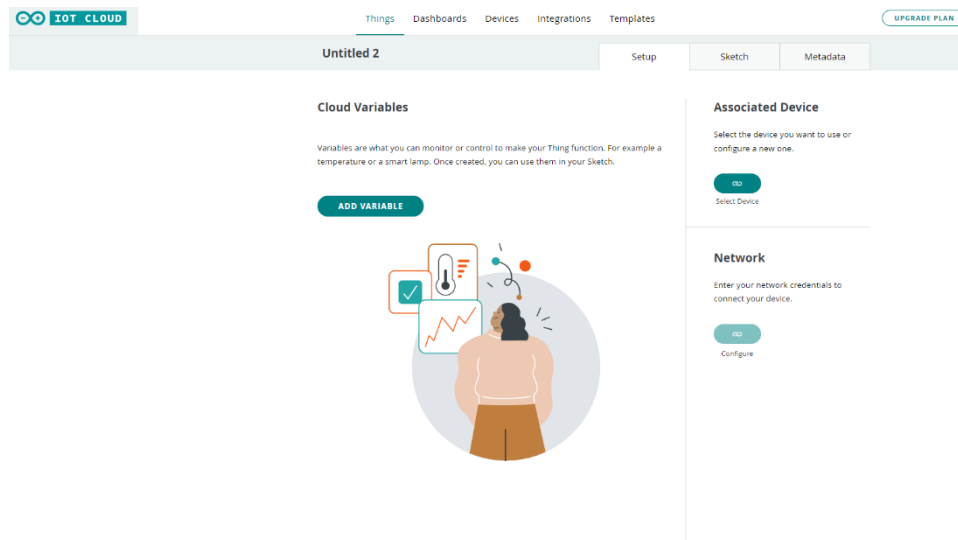


Figura N.º 4. Pantalla inicial de objetos.

- **Creación de Variables:** Se crean las variables *humidity* y *temperature* para el la recolección de la data y la variable *message* para notificar e imprimir en consola los valores recolectados.

Add variable
×

Name
|

↺ Sync with other Things ⓘ

Select variable type ▼

Declaration ⓘ

Variable Permission ⓘ

☒ Read & Write
☐ Read Only

Variable Update Policy ⓘ

☒ On change
☐ Periodically

Threshold
0

ADD VARIABLE CANCEL

Figura N.º 5. Creación de variables.

Cloud Variables
ADD

	Name ↓	Last Value	Last Update	
<input type="checkbox"/>	humidity float humidity;	63	08 Nov 2023 19:09:36	⋮
<input type="checkbox"/>	message string message;	Temperature= 24.00...	08 Nov 2023 19:09:39	⋮
<input type="checkbox"/>	temperature float temperature;	23.9	08 Nov 2023 19:09:36	⋮

Figura N.º 6. Variables principales del trabajo.

- Creación de Device:** Se conecta el hardware a través de la configuración de *Device* seleccionando ESP32 y el tipo NodeMCU-32S. Luego se configura la red Wifi a la que se interconectarán especificando nombre de red, contraseña y la *Key* que previamente se obtuvo de la creación del *Device*.

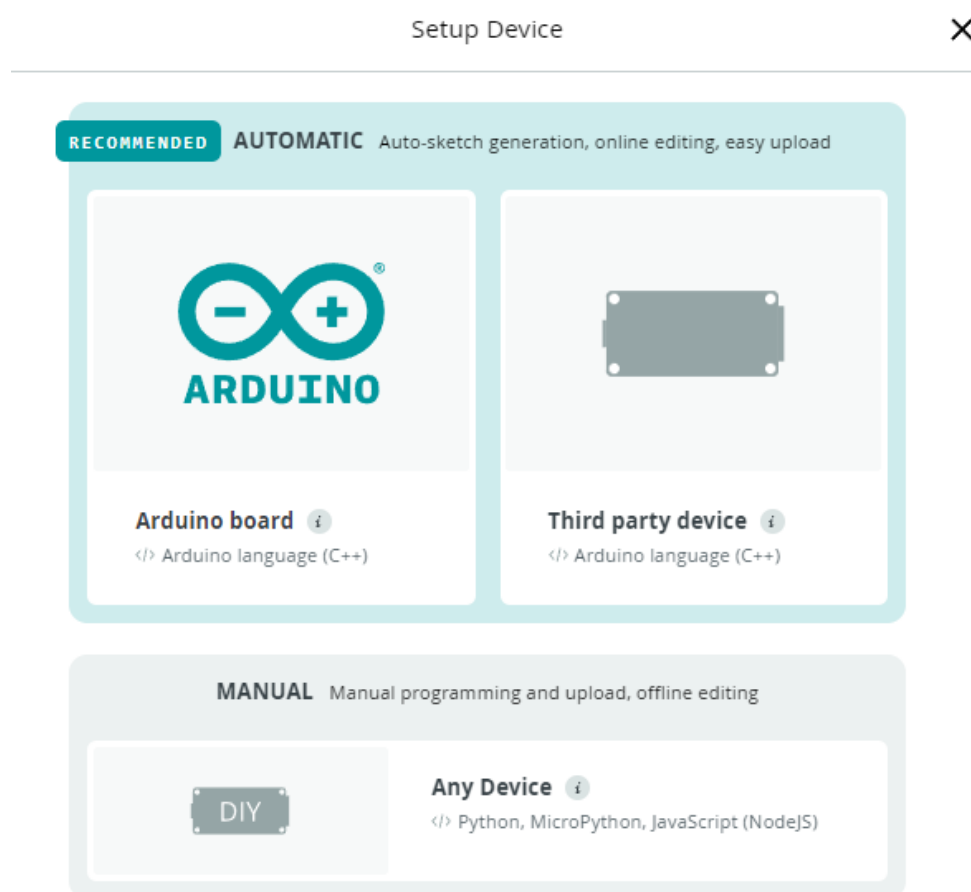


Figura N.º 7. Configuración del dispositivo.

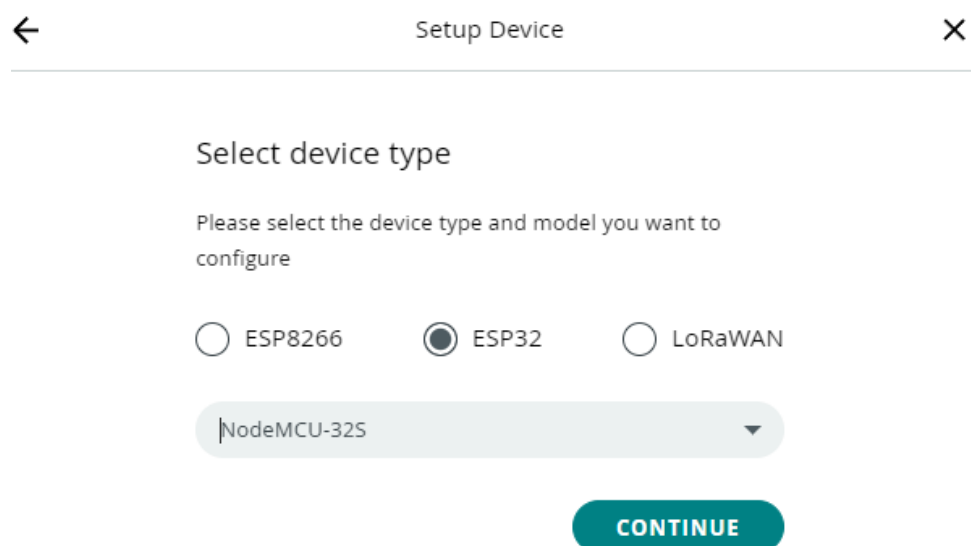



Figura N.º 8. Selección de ESP32.

Associated Device



Esp32

ID: 45bb1076-f445-4ac2-a725-... 

Type: NodeMCU-32S

Status:  Offline



Change



Detach

Figura N.º 9. Dispositivo vinculado.

- Configuración de Red:

Network

Enter your network credentials to
connect your device.



Configure

Figura N.º 10. Configuración de red.

Configure network ✕

You will find these network parameters in the secret tab in your sketch, and your device will be able to connect to the network once the sketch will be uploaded.

Wi-Fi Name *

Botsito

Password

.....

👁

Secret Key *


.....


👁

SAVE

Figura N.º 11. Parámetros de la red.

- **Dashboard Arduino IOT Cloud**
 - **Creación de Dashboard:** En la pestaña *Dashboard* , se crea un nuevo entorno donde principalmente se añadirán los elementos *Gauge* (medidor) y *Chart* (gráfica) que mostrará el valor actual recolectado y lo registrará en un gráfico de líneas respectivamente.

 Dashboards

 Get help

CREATE

Figura N.º 12. Pestaña de Dashboards.

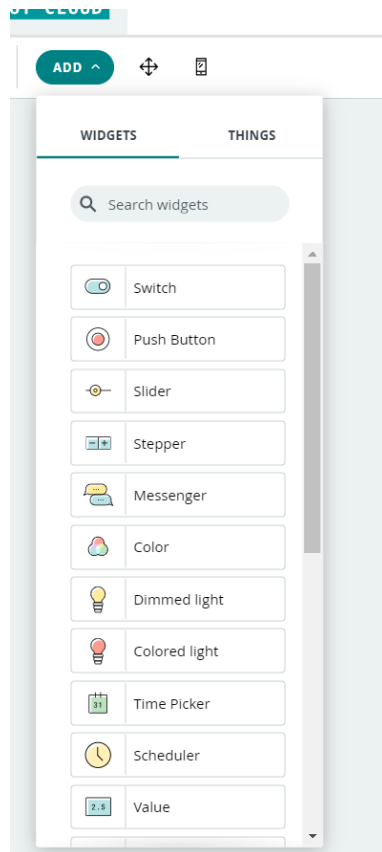


Figura N.º 13. Elementos a añadir al Dashboard.

- **Asociación de Dashboard con variables:** Tanto la variable *humidity* (humedad) como la de *temperature* (temperatura) se asocian dentro de los *Gauges,s* y *Chart,s*

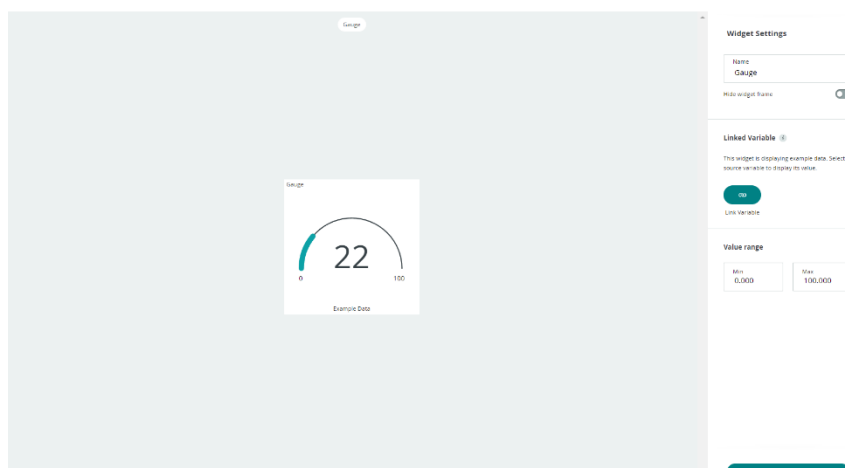


Figura N.º 14. Medidor de variable humedad.

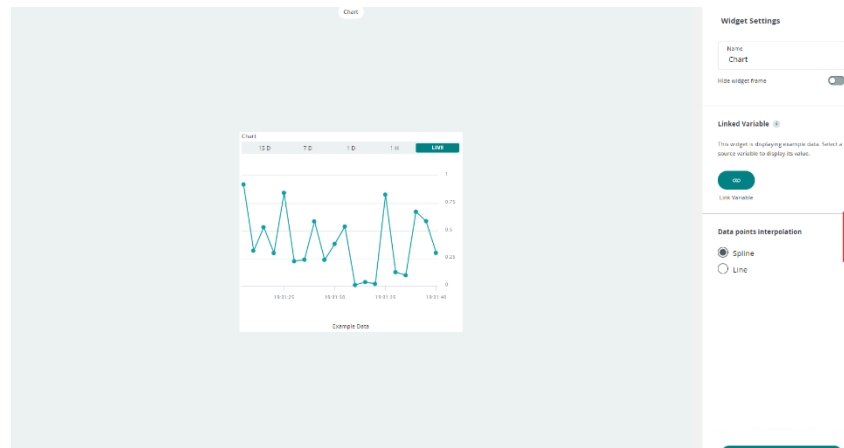


Figura N.º 15. Gráfica de variable humedad.

- **Dashboard en tiempo real:** Dashboard final con la data recopilada de humeada y temperatura.



Figura N.º 16. Dashboard final de las variables temperatura y humedad.

- **Script de Python para manejo de datos:** Se inicializa la librería Pandas y se leen los archivos csv provenientes del Arduino

```
#Libreria
import pandas as pd

#TEMPERATURA
# Lee el primer archivo CSV
tmp1 = pd.read_csv('C:\\Users\\acer\\Pictures\\7mo CICLO\\INTERNET DE LAS
COSAS\\TAREA2\\Untitled-temperature.csv')

# Lee el segundo archivo CSV
tmp2 = pd.read_csv('C:\\Users\\acer\\Pictures\\7mo CICLO\\INTERNET DE LAS
COSAS\\TAREA2\\Untitled-temperature(1).csv')

#HUMEDAD
# Lee el primer archivo CSV
hmd1 = pd.read_csv('C:\\Users\\acer\\Pictures\\7mo CICLO\\INTERNET DE LAS
COSAS\\TAREA2\\Untitled-humidity.csv')

# Lee el segundo archivo CSV
hmd2 = pd.read_csv('C:\\Users\\acer\\Pictures\\7mo CICLO\\INTERNET DE LAS
COSAS\\TAREA2\\Untitled-humidity(1).csv')
.
.
.
#N archivos
```

Convertimos la columna de *time* (fecha) a un formato más entendible y reducimos a 1 decimal la variable temperatura la columna de *value*.

```
#TEMPERATURA
# Convierte la columna 'time' en un objeto de fecha y hora
tmp1['time'] = pd.to_datetime(tmp1['time'], format='%Y-%m-%dT%H:%M:%S.%fZ')

# Redondea la columna 'value' a 2 decimales
tmp1['value'] = tmp1['value'].round(2)

# Convierte la columna 'time' en un objeto de fecha y hora
tmp2['time'] = pd.to_datetime(tmp2['time'], format='%Y-%m-%dT%H:%M:%S.%fZ')

# Redondea la columna 'value' a 2 decimales
tmp2['value'] = tmp2['value'].round(2)

#HUMEDAD
# Convierte la columna 'time' en un objeto de fecha y hora
hmd1['time'] = pd.to_datetime(hmd1['time'], format='%Y-%m-%dT%H:%M:%S.%fZ')

# Convierte la columna 'time' en un objeto de fecha y hora
hmd2['time'] = pd.to_datetime(hmd2['time'], format='%Y-%m-%dT%H:%M:%S.%fZ')
```


Combinamos ambos csv en uno solo.

```
#TEMPERATURA
# Combina ambos DataFrames en uno solo
df_combinedt = pd.concat([tmp1, tmp2], ignore_index=True)

# Guarda los datos procesados y combinados en un nuevo archivo CSV
df_combinedt.to_csv('C:\\Users\\acer\\Pictures\\7mo CICLO\\INTERNET DE LAS
COSAS\\TAREA2\\TEMPERATURA.csv', index=False)

#HUMEDAD
# Combina ambos DataFrames en uno solo
df_combinedh = pd.concat([hmd1, hmd2], ignore_index=True)

# Guarda los datos procesados y combinados en un nuevo archivo CSV
df_combinedh.to_csv('C:\\Users\\acer\\Pictures\\7mo CICLO\\INTERNET DE LAS
COSAS\\TAREA2\\HUMEDAD.csv', index=False)
```

Añadimos la columna de humedad al csv resultante, donde tengan la misma fecha y hora, para comprimir toda la data en un solo archivo resultante. Eliminamos duplicados.

```
df_combinedt = df_combinedt.rename(columns={'time': 'fecha y hora', 'value':
'temperatura'})

df_combinedh = df_combinedh.rename(columns={'time': 'fecha y hora', 'value': 'humedad'})

# Combina ambos DataFrames utilizando la columna 'time' como clave
df_final = pd.merge(df_combinedt, df_combinedh[['fecha y hora', 'humedad']], on='fecha y
hora', how='left')
df_final_nd = df_final.drop_duplicates()

# Guarda el resultado en un archivo CSV final
df_final_nd.to_csv('C:\\Users\\acer\\Pictures\\7mo CICLO\\INTERNET DE LAS
COSAS\\TAREA2\\DATA_TEMPERATURA_y_HUMEDAD.csv', index=False)
```

3. Resultados

Durante las pruebas, se llevaron a cabo mediciones de humedad y temperatura con el sensor. Se realizaron múltiples mediciones en diferentes condiciones para evaluar la precisión y la fiabilidad del sensor. A continuación, se resumen los resultados principales:

- **Prueba inicial (Arduino IDE):** Previo a la prueba en el entorno de Arduino IOT Cloud, se realizó las pruebas del código y el hardware en una primera compilación en Arduino IDE. Resultados mostrados a continuación:

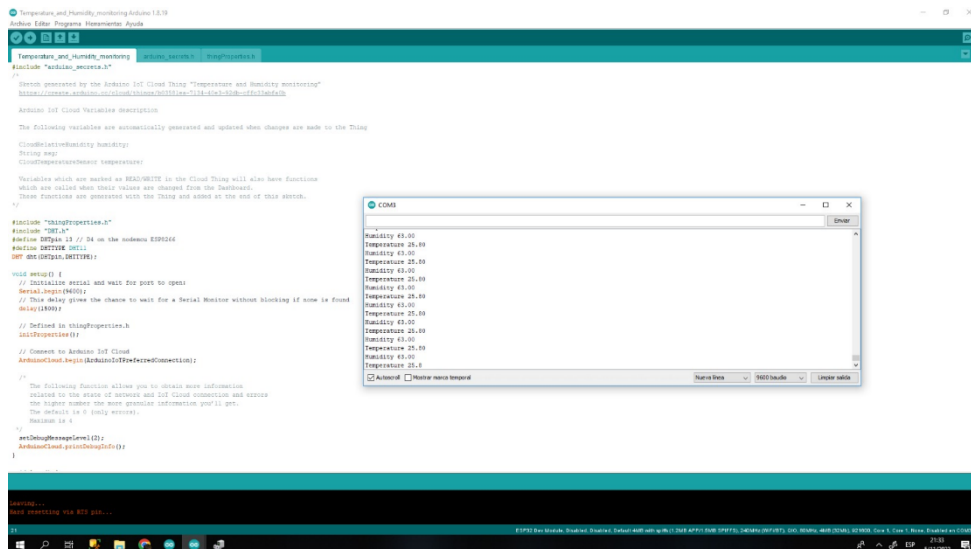


Figura N.º 17. Output Arduino IDE

- **Prueba final (Arduino IOT Cloud):** Lo componen las variables Humedad (*humidity*) , Temperatura (*temperature*) y Mensaje (*message*); junto a ella está la configuración del *Device* o dispositivo ESP32 y la configuración de la red. Luego se tiene el código (ya utilizado en el Sketch) en Arduino detallado en su configuración.

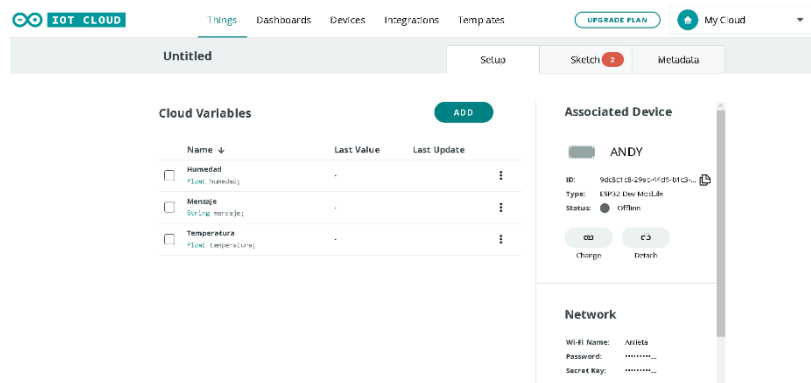


Figura N.º 18. Vista general de Things del Arduino IOT Cloud

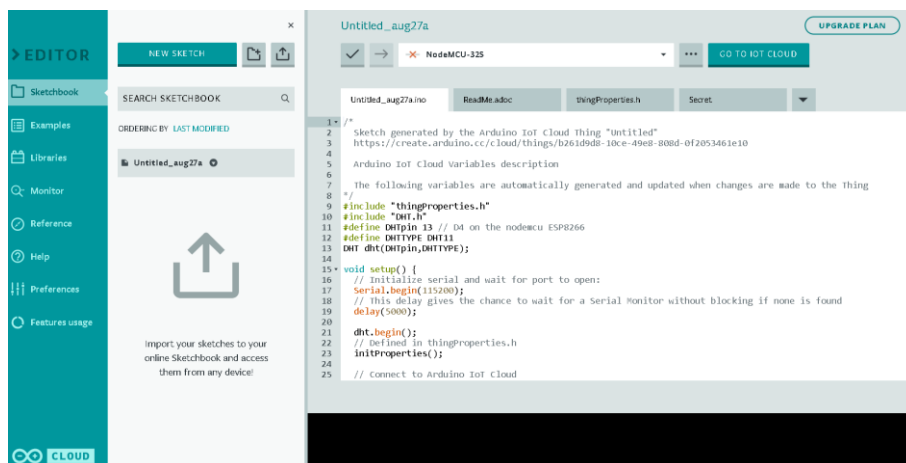


Figura N.º 19. Vista de Sketch de código en Arduino IOT Cloud

- **Archivos CSV:** Archivos resultantes del entregable.

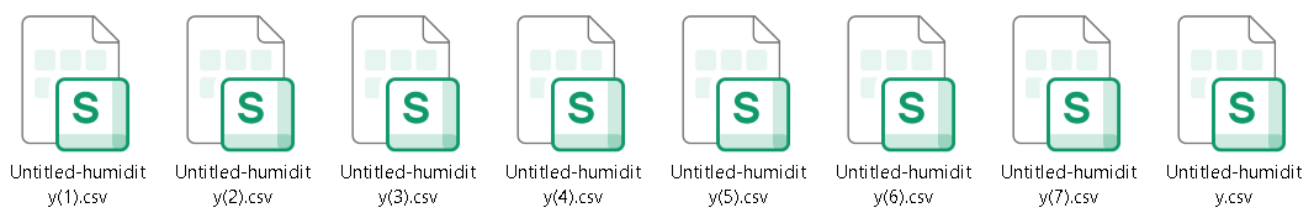


Figura N.º 20. Archivos resultantes del proceso para la humedad.

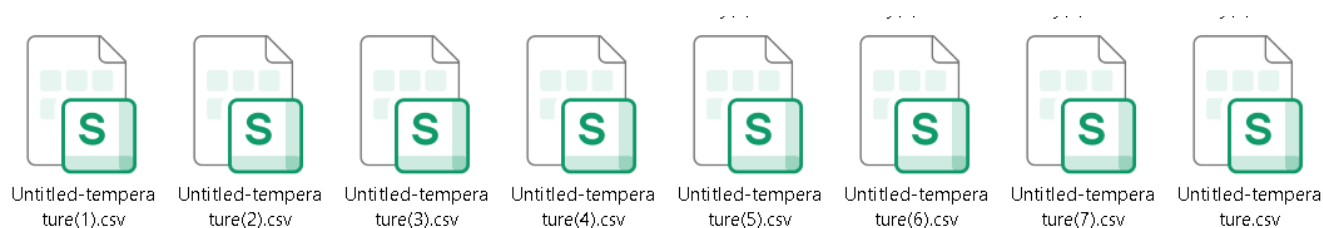


Figura N.º 21. Archivos resultantes del proceso para la temperatura.



HUME.csv



TEMP.csv

Figura N.º 22. Archivos consintenciados para la unión.



DATA_TEMP_y_H
UME_G1.csv

Figura N.º 23. Archivo final del entregable.

A1		fecha y hora,temperatura,humedad					
	A	B	C	D	E	F	G
1	fecha y hora,temperatura,humedad						
2	2023-11-07 04:40:58.495638584,22.6,67						
3	2023-11-07 04:45:22.162484770,25.2,89						
4	2023-11-07 04:46:09.176283175,25.6,86						
5	2023-11-07 04:46:09.266258334,25.7,85						
6	2023-11-07 04:46:09.460580286,26.2,87						
7	2023-11-07 04:46:23.160042032,26.7,88						
8	2023-11-07 04:46:23.161727200,27.1,89						
9	2023-11-07 04:46:23.422570153,27.5,91						
10	2023-11-07 04:46:23.490553292,27.9,92						
11	2023-11-07 04:46:35.302266418,28.0,90						
12	2023-11-07 04:46:45.312647186,27.7,87						
13	2023-11-07 04:46:51.460274005,27.6,86						
14	2023-11-07 04:46:53.620241562,27.5,84						
15	2023-11-07 04:46:57.354352863,27.4,82						
16	2023-11-07 04:47:01.332427819,27.3,80						
17	2023-11-07 04:47:05.328317399,27.2,77						
18	2023-11-07 04:47:13.526145947,27.1,75						
19	2023-11-07 04:47:15.406392162,27.0,71						
20	2023-11-07 04:47:21.492748315,26.9,68						
21	2023-11-07 04:47:25.394335549,26.8,66						
22	2023-11-07 04:47:29.348471280,26.7,64						
23	2023-11-07 04:47:33.458357816,26.6,63						
24	2023-11-07 04:47:45.418209222,26.4,61						
25	2023-11-07 04:52:46.526308031,25.0,63						
26	2023-11-07 04:52:49.298482760,24.3,64						
27	2023-11-07 04:57:37.706919574,23.8,67						
28	2023-11-07 04:57:41.702887361,23.9,69						
29	2023-11-07 05:13:43.046805557,23.6,67						
K < > >		DATA TEMP y HUME G1			+		

Figura N.º 20. Vista de archivo final del entregable.

4. Conclusiones

Se tiene que el uso de un script de Python permitió un manejo eficiente de los datos adquiridos. Se logró convertir la columna de fechas en un formato más legible y reducir la variable de temperatura a un decimal, lo que facilita la comprensión y el análisis de los datos.

La consolidación de los datos de humedad y temperatura en un solo archivo resultante fue exitosa, lo que simplifica la gestión de la información y permite un análisis más completo al tener ambos conjuntos de datos en un solo lugar.

Antes de la implementación en el entorno de Arduino IoT Cloud, se realizaron pruebas iniciales en el Arduino IDE para verificar la funcionalidad del código y el hardware. Estas pruebas permitieron asegurar que el sensor DHT y el dispositivo Arduino estuvieran funcionando correctamente.

La implementación exitosa del código en Arduino IoT Cloud permitió la recopilación continua de datos de humedad y temperatura en un entorno en la nube, lo que facilita el monitoreo y la gestión a distancia de los datos.

La plataforma Arduino IoT Cloud ofrece una interfaz intuitiva para visualizar y gestionar los datos recopilados. Se pueden observar gráficos de humedad y temperatura en tiempo real, lo que facilita el seguimiento de condiciones ambientales.

Los archivos CSV resultantes del proceso inicial se mejoraron con el script de Python, lo que facilita la interpretación y el análisis de los datos por parte de los usuarios.

En resumen, el proyecto logró implementar un sistema de adquisición de datos de humedad y temperatura a través de un sensor DHT, procesar eficazmente estos datos utilizando un script de Python y presentarlos de manera más legible en archivos CSV. La transición a Arduino IoT Cloud permitió una gestión más efectiva de los datos, y la eliminación de datos duplicados garantizó su calidad. Estos resultados sientan las bases para futuros análisis y aplicaciones en el ámbito de la monitorización ambiental y la automatización.

5. Referencias

- Tutoriales, C. (s. f.). *Detector de movimiento con luz, ESP32 y notificaciones por Telegram – RogerBit*. <https://rogerbit.com/wprb/2021/01/detector-de-movimiento-con-luz-esp32-y-notificaciones-por-telegram/>
- Elosiloscopio. (2021, 21 julio). HC-SR501 Sensor de movimiento PIR para arduino, ESP8266 y ESP32. *El Osciloscopio*. <https://elosiloscopio.com/hc-sr501-sensor-de-movimiento-pir-para-arduino-esp8266-y-esp32/>