

***"Año de la recuperación y consolidación de la economía peruana"***



**Servicio Nacional de Adiestramiento en Trabajo Industrial**

**Alumno:** Montero Mandujano Victor Junior.

**Carrera:** Ingeniería de Software con Inteligencia Artificial.

**Semestre:** 5to.

**Sede:** Huánuco.

**Instructor:** Anthony Heli Barra Espinoza.

**Modulo:** TALLER DE DESARROLLO DE APLICACIONES CON MACHINE LEARNING.

**Proyecto:** Tarea09.

**2025**

## **Introducción**

En esta práctica se desarrolló un sistema inteligente de monitoreo de temperatura aplicando herramientas de Machine Learning y TinyML. El objetivo principal fue crear un modelo que pueda detectar cuando la temperatura supera los límites seguros en un ambiente de almacenamiento, y generar una alerta visual y sonora para prevenir daños en productos perecibles.

El trabajo combina el uso de TensorFlow para el entrenamiento del modelo, TensorFlow Lite para su optimización, y SimulIDE para la simulación del comportamiento del sistema con sensores, LED y buzzer. De esta manera, se demuestra cómo la inteligencia artificial puede integrarse en sistemas embebidos de bajo costo, manteniendo una respuesta rápida y eficiente incluso sin conexión a internet.

## Desarrollo del Proyecto

### ◆ Etapas principales realizadas

#### 1. Entrenamiento del modelo en Google Colab:

Se utilizó TensorFlow para entrenar un modelo que aprende la conversión entre grados Celsius y Fahrenheit.

#### 2. Conversión a TensorFlow Lite:

El modelo entrenado (modelo\_temperatura.keras) se convirtió a un formato más ligero (modelo\_temperatura.tflite) para poder ejecutarlo en dispositivos de bajo consumo.

#### 3. Generación del modelo TinyML (C/C++):

El archivo .tflite fue convertido en código C (model.h) mediante un script en Google Colab, para integrarse dentro de un proyecto Arduino real.

#### 4. Simulación en SimulIDE:

Se utilizó el sensor DHT11 para simular lecturas de temperatura.

Si la temperatura supera los 25 °C, se activa una alarma sonora (buzzer) y un LED de advertencia.

### ◆ Pasos para implementación en hardware real (TinyML)

#### 1. Entrenar el modelo en TensorFlow y convertirlo a formato .tflite:

```
1 converter = tf.lite.TFLiteConverter.from_keras_model(model)
2 tflite_model = converter.convert()
3 open('modelo_temperatura.tflite', 'wb').write(tflite_model)
```

#### 2. Convertir el archivo .tflite a formato C (model.h):

```
5 data = open('modelo_temperatura.tflite', 'rb').read()
6 with open('model.h', 'w') as f:
7     f.write('#pragma once\n#include <stdint.h>\n')
8     f.write('const unsigned char g_model[] = {')
9     for i, b in enumerate(data):
10         if i % 12 == 0: f.write('\n ')
11         f.write(f'0x{b:02x}, ')
12     f.write('\n};\n')
13     f.write(f'const unsigned int g_model_len = {len(data)};\n')
14 print("Archivo model.h generado (TinyML listo)")
15
```

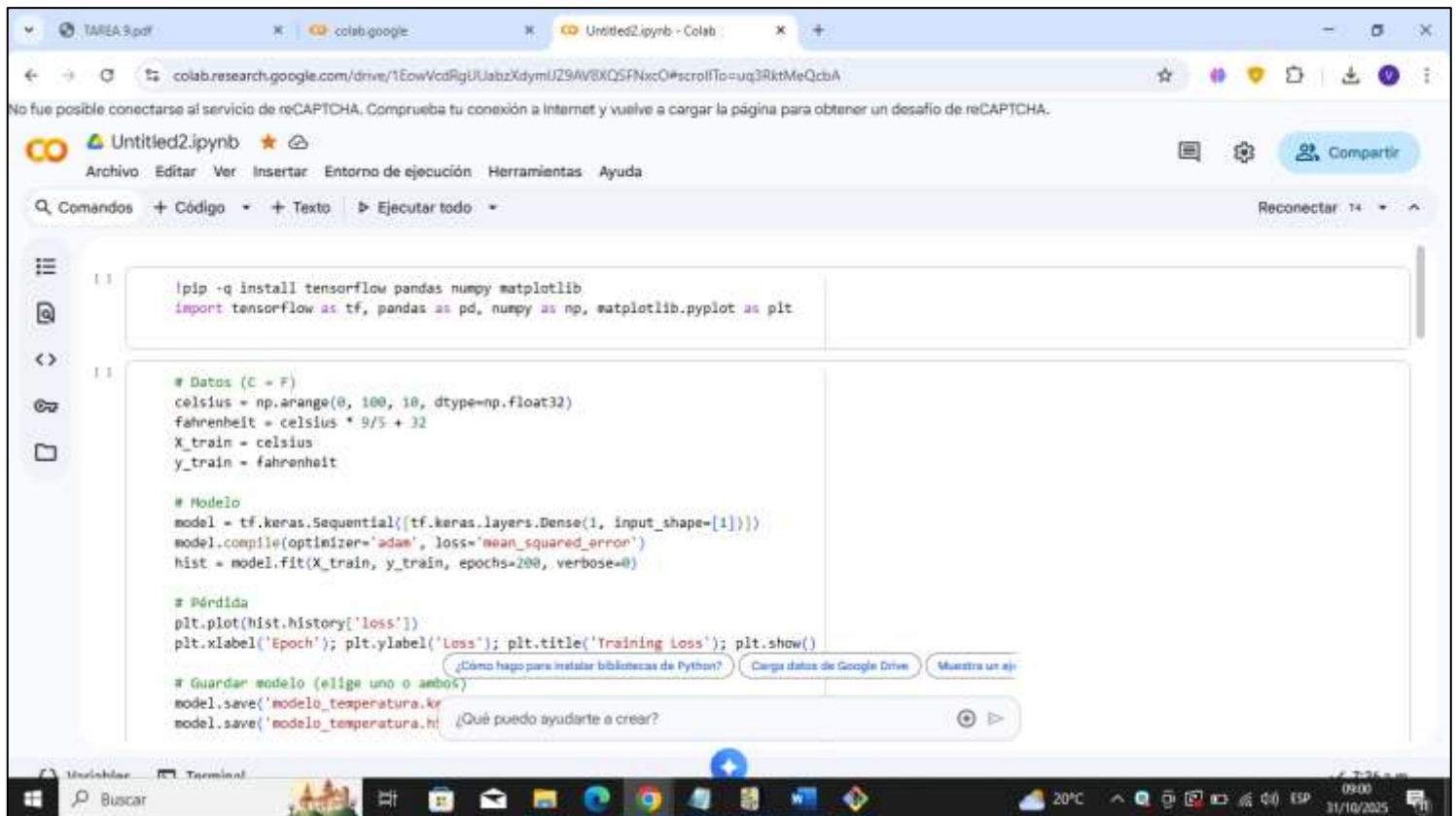
**3.** Incluir el model.h dentro de un sketch de Arduino (tinymL\_alarma.ino):

- Este sketch combina el modelo TinyML con el control del sensor DHT11, el LED y el buzzer.
- Así, el microcontrolador puede predecir y actuar localmente, sin depender de una computadora externa.

**4.** Cargar el código y el modelo al microcontrolador (Arduino Nano 33 BLE o ESP32).

El sistema detectará el aumento de temperatura y encenderá el LED y buzzer automáticamente.

## Códigos para la creación de los archivos TensorFlow, TensorFlow Lite y model.h (TinyML) ejecutados en Google Colab y que se implementarán en el proyecto



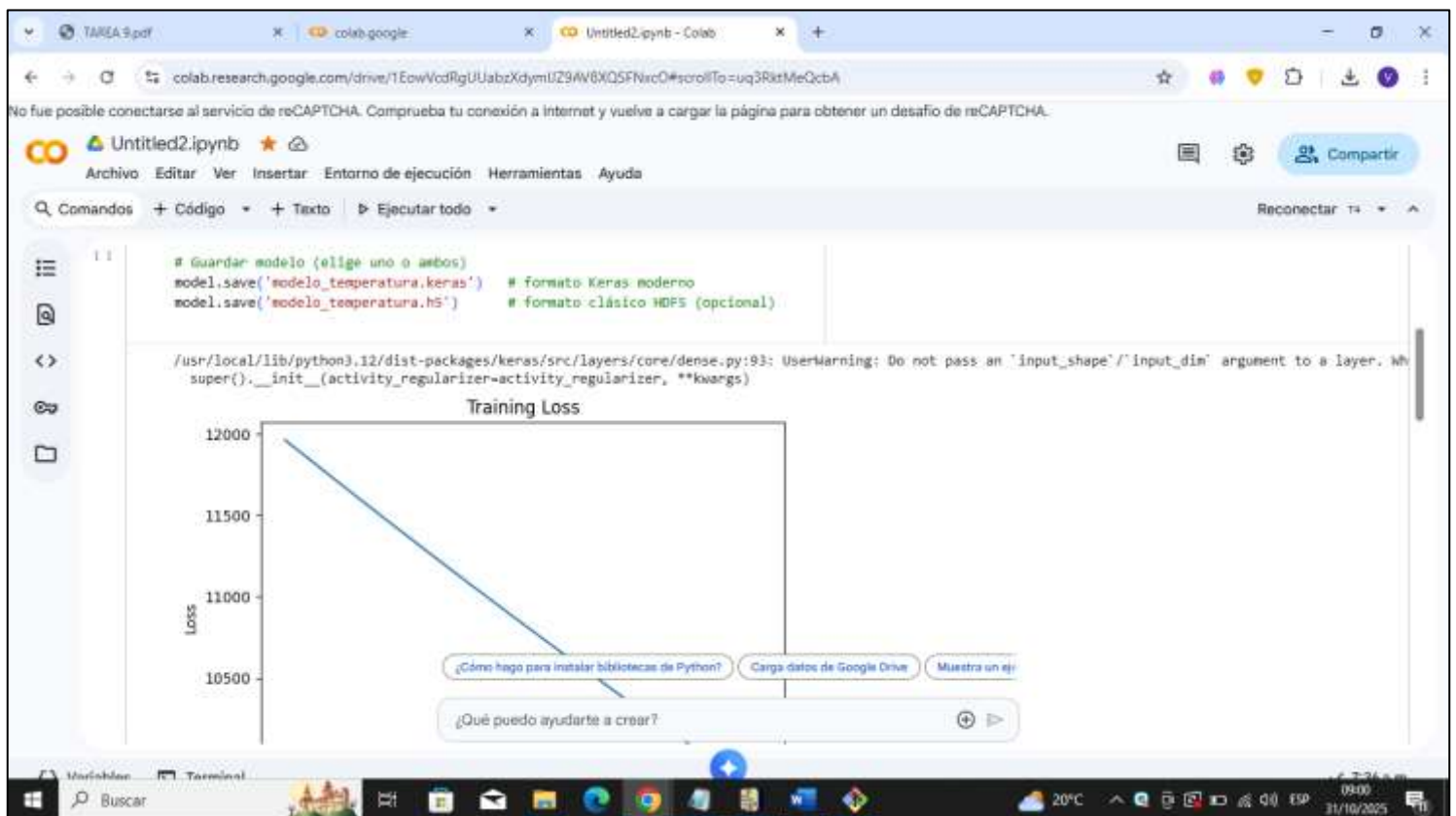
```
1 | !pip -q install tensorflow pandas numpy matplotlib
   | import tensorflow as tf, pandas as pd, numpy as np, matplotlib.pyplot as plt

2 | # Datos (C = F)
   | celsius = np.arange(0, 100, 10, dtype=np.float32)
   | fahrenheit = celsius * 9/5 + 32
   | X_train = celsius
   | y_train = fahrenheit

3 | # Modelo
   | model = tf.keras.Sequential([tf.keras.layers.Dense(1, input_shape=[1])])
   | model.compile(optimizer='adam', loss='mean_squared_error')
   | hist = model.fit(X_train, y_train, epochs=200, verbose=0)

4 | # Pérdida
   | plt.plot(hist.history['loss'])
   | plt.xlabel('Epoch'); plt.ylabel('Loss'); plt.title('Training Loss'); plt.show()

5 | # Guardar modelo (elige uno o ambos)
   | model.save('modelo_temperatura.keras')
   | model.save('modelo_temperatura.h5')
```



TAREA 9.pdf x colab.google x Untitled2.ipynb - Colab x +

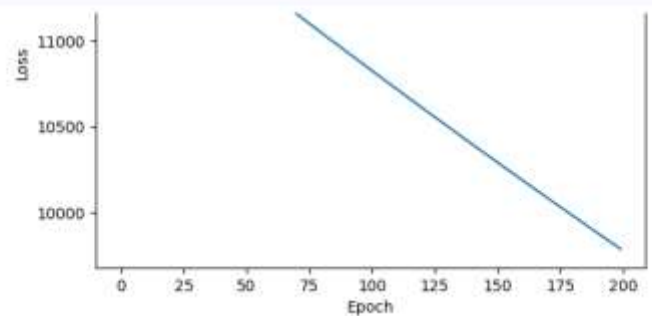
colab.research.google.com/drive/TEowVdRgUUabzXdymlIZ9AV8XQ5FNxcO#scrollTo=uq3RktMeQcbA

No fue posible conectarse al servicio de reCAPTCHA. Comprueba tu conexión a Internet y vuelve a cargar la página para obtener un desafío de reCAPTCHA.

Untitled2.ipynb

Archivo Editar Ver Insertar Entorno de ejecución Herramientas Ayuda

Comandos + Código + Texto ▶ Ejecutar todo + Reconectar T4 + ^



WARNING:absl:You are saving your model as an HDF5 file via 'model.save()' or 'keras.save\_model(model)'. This file format is considered legacy. We

```
11 converter = tf.lite.TFLiteConverter.from_keras_model(model)
12 tflite_model = converter.convert()
13 open('modelo_temperatura.tflite', 'wb').write(tflite_model)
14 print('Listo: modelo_temperatura.tflite')
```

¿Qué puedo ayudarte a crear?

TAREA 9.pdf x colab.google x Untitled2.ipynb - Colab x +

colab.research.google.com/drive/TEowVdRgUUabzXdymlIZ9AV8XQ5FNxcO#scrollTo=uq3RktMeQcbA

No fue posible conectarse al servicio de reCAPTCHA. Comprueba tu conexión a Internet y vuelve a cargar la página para obtener un desafío de reCAPTCHA.

Untitled2.ipynb

Archivo Editar Ver Insertar Entorno de ejecución Herramientas Ayuda

Comandos + Código + Texto ▶ Ejecutar todo + Reconectar T4 + ^

```
11 tflite_model = converter.convert()
12 open('modelo_temperatura.tflite', 'wb').write(tflite_model)
13 print('Listo: modelo_temperatura.keras (o .h5) y modelo_temperatura.tflite')
```

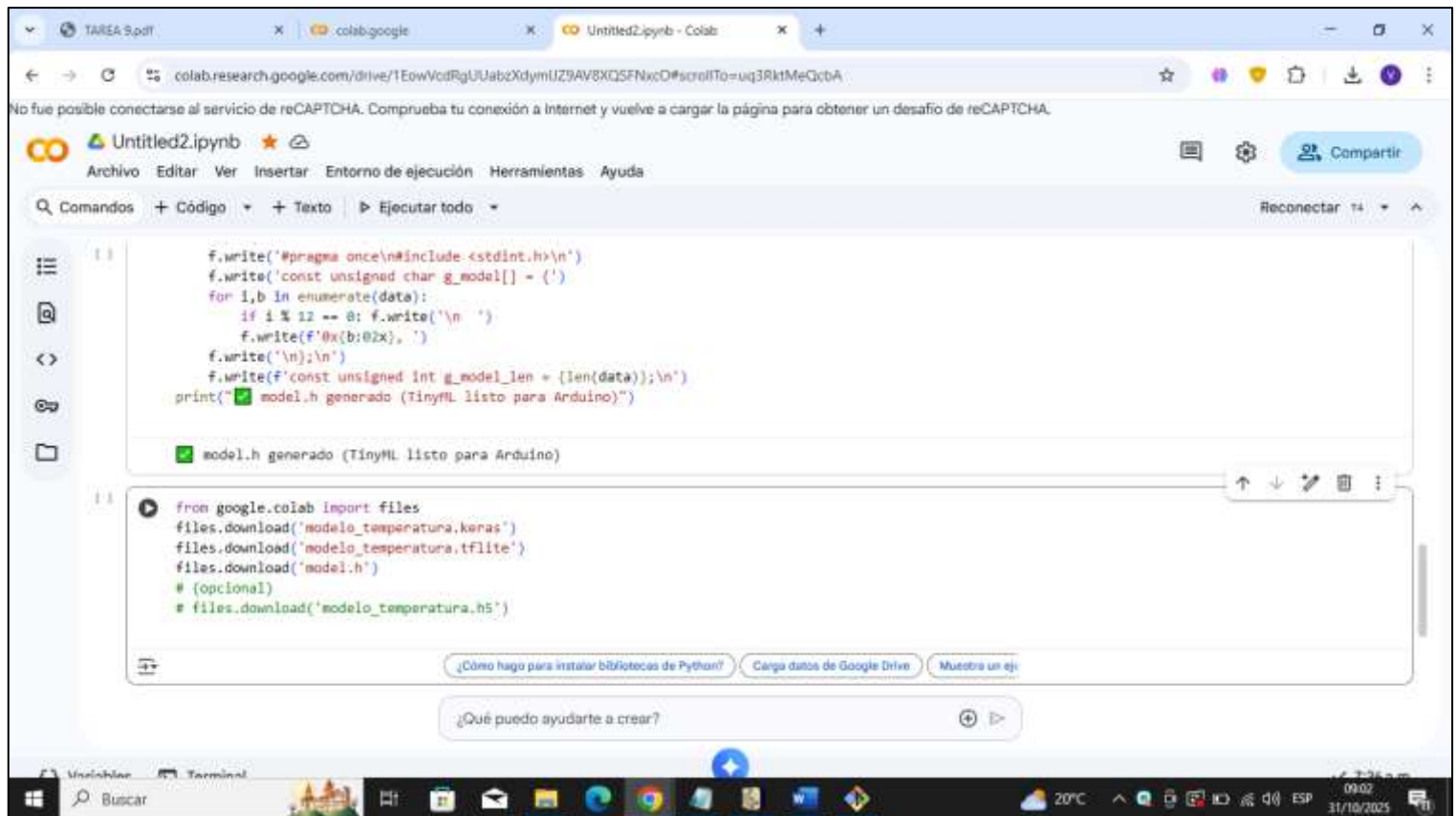
Saved artifact at '/tmp/tmpFb7\_\_ws'. The following endpoints are available:

- \* Endpoint 'serve'
- args\_B (POSITIONAL\_ONLY): TensorSpec(shape=(None, 1), dtype=tf.float32, name='keras\_tensor')
- Output Type: TensorSpec(shape=(None, 1), dtype=tf.float32, name=None)
- Captures: 139983341432144: TensorSpec(shape=(), dtype=tf.resource, name=None) 139983341433296: TensorSpec(shape=(), dtype=tf.resource, name=None)
- Listo: modelo\_temperatura.keras (o .h5) y modelo\_temperatura.tflite

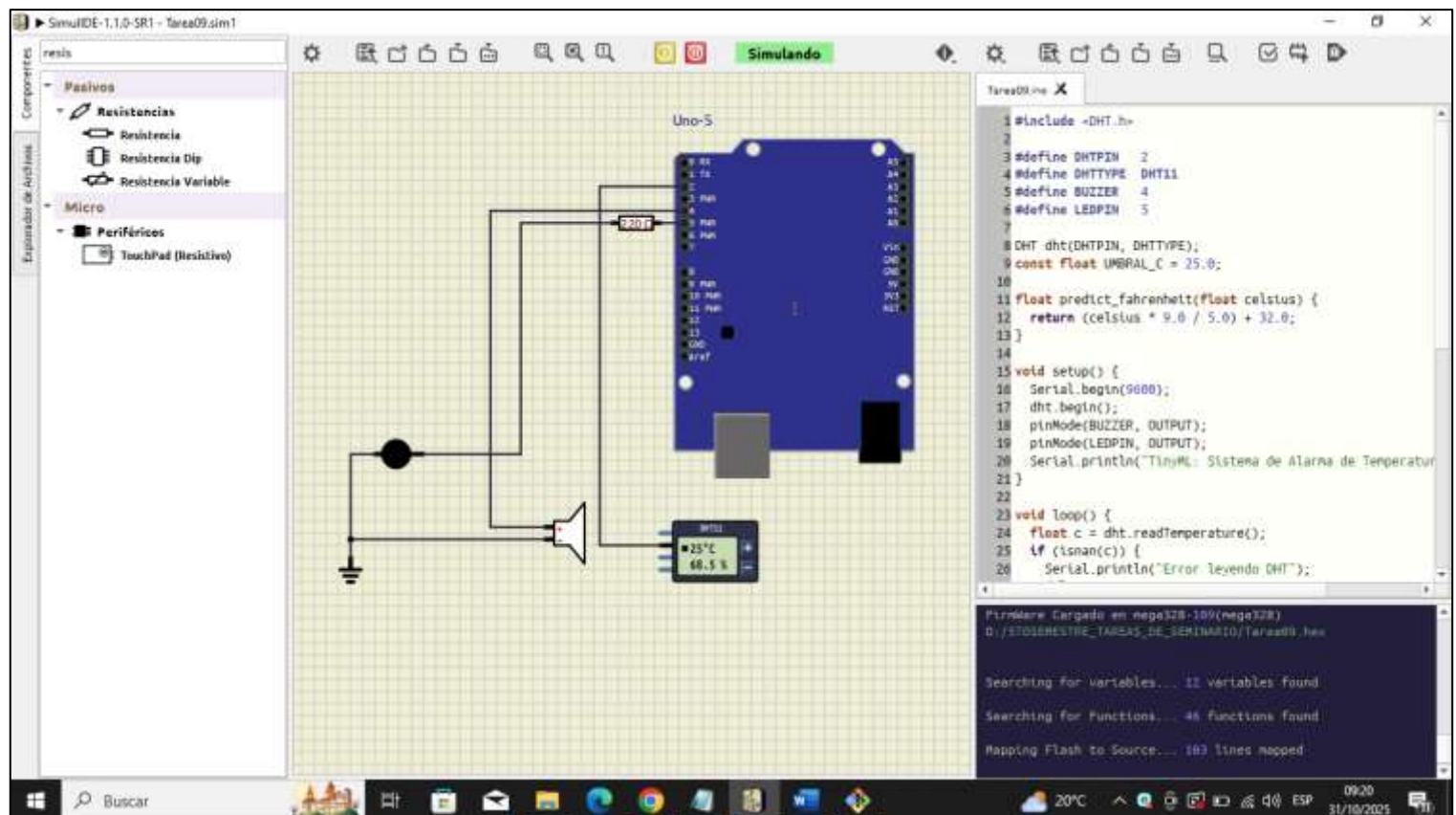
```
11 # Convierte el .tflite a un header C para usar en Arduino (tinyML)
12 data = open('modelo_temperatura.tflite', 'rb').read()
13 with open('model.h', 'w') as f:
14     f.write('#pragma once\n#include <stdint.h>\n')
15     f.write('const unsigned char model_data[] = {')
16     for i, b in enumerate(data):
17         if i % 12 == 0: f.write('\n    ')
18         f.write(f'0x{b:02x}, ')
19     f.write('};')
```

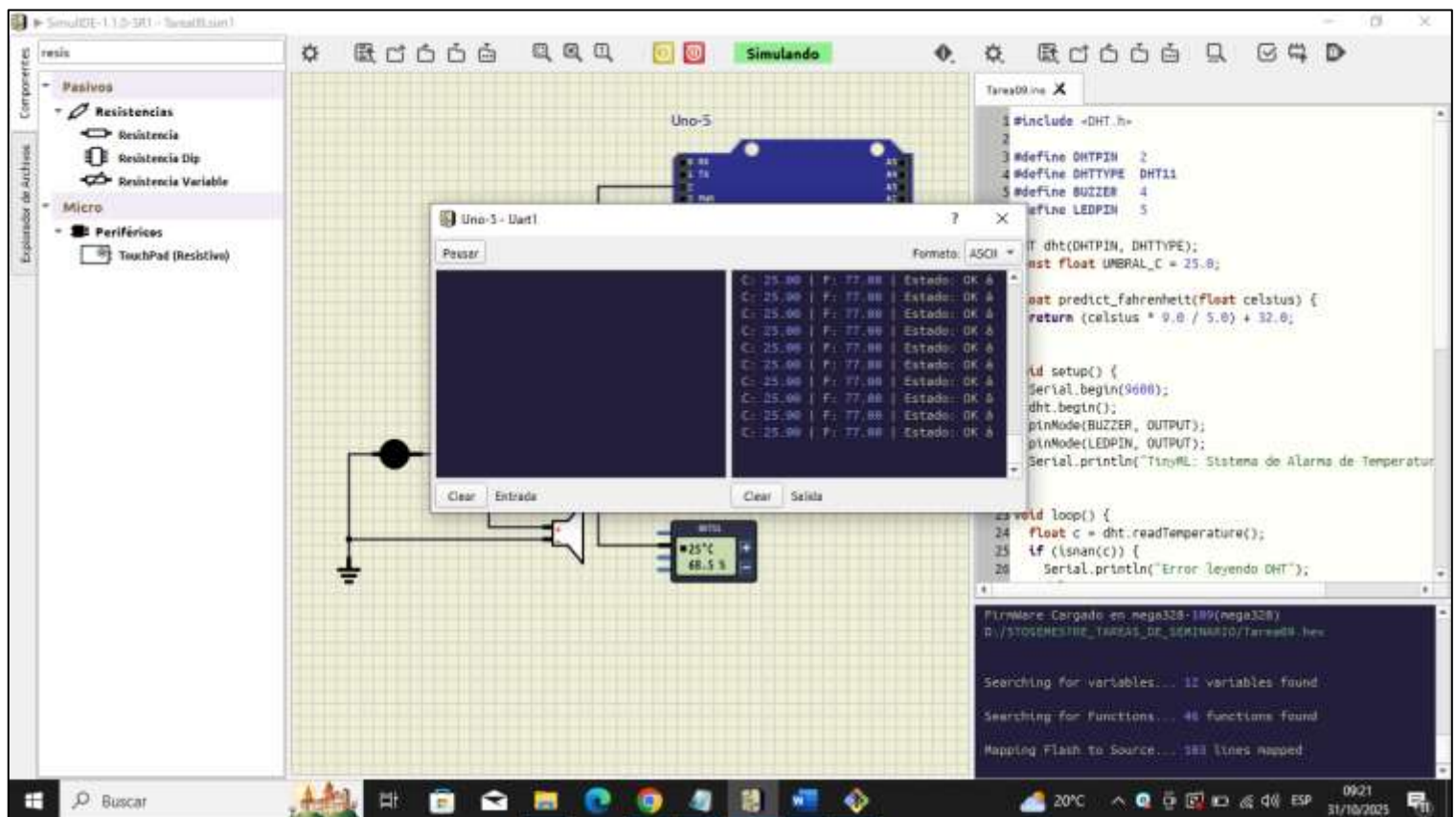
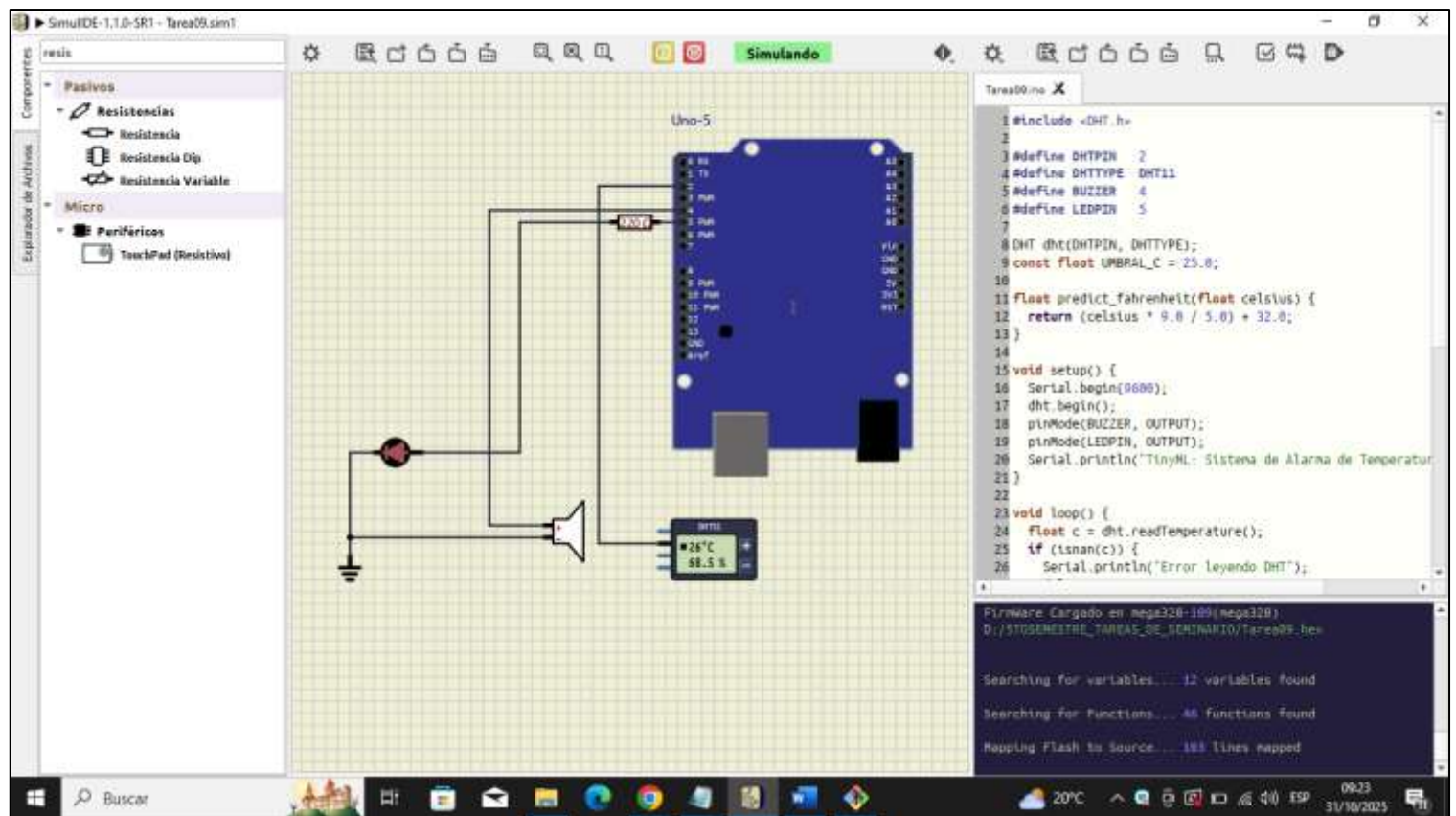
¿Qué puedo ayudarte a crear?



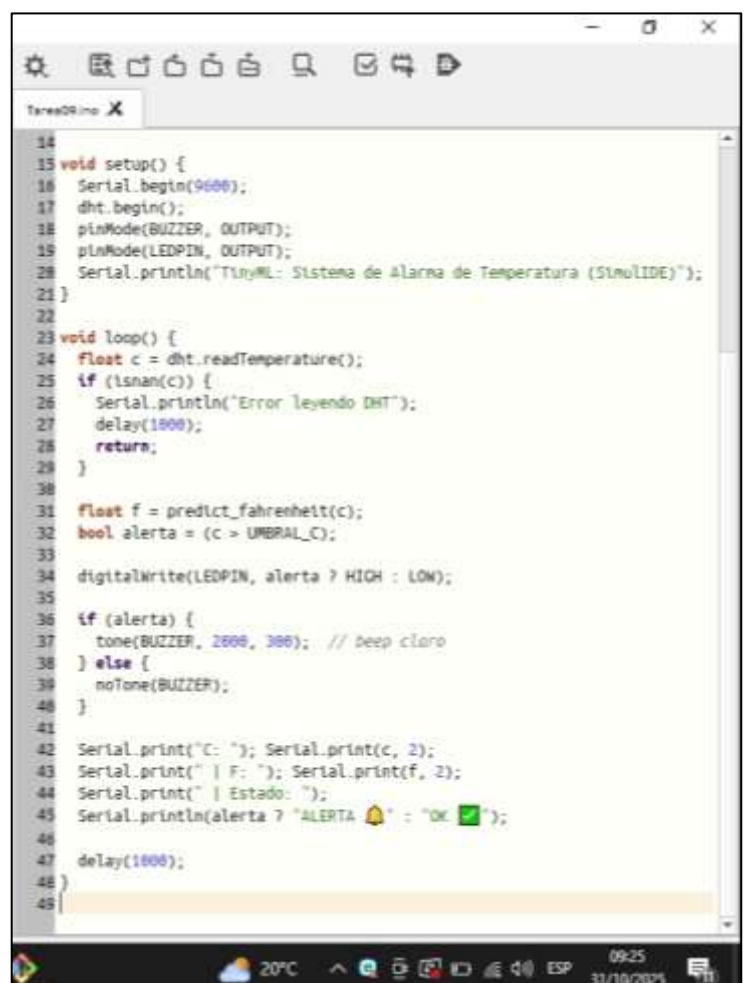
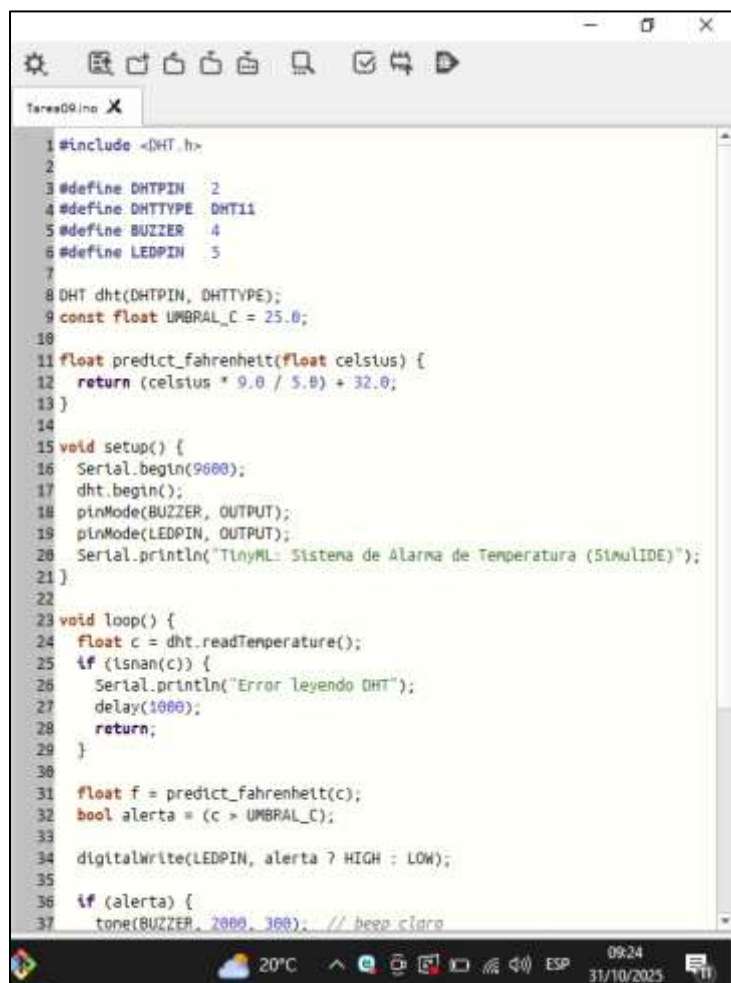
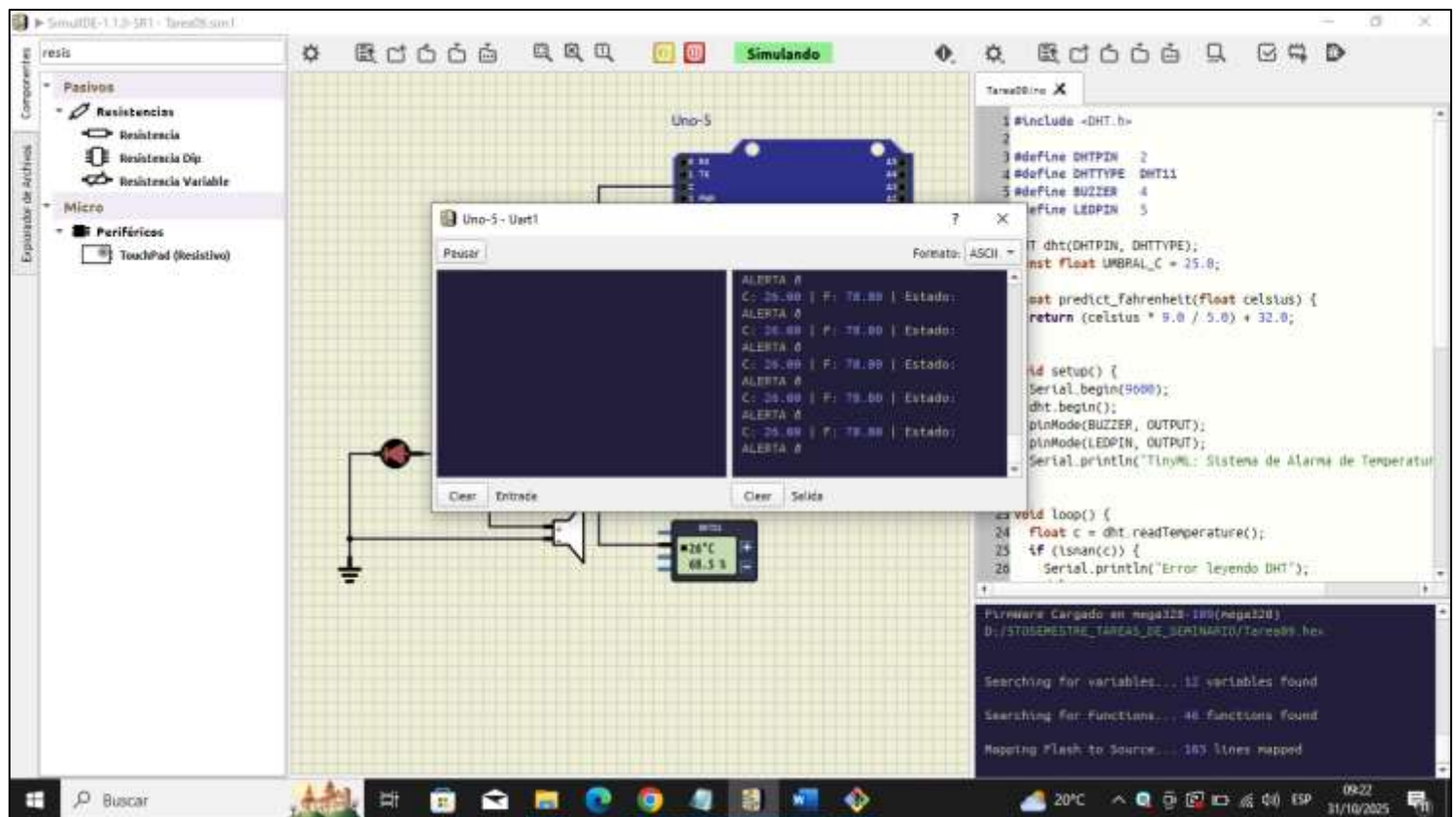


Circuito .sim1(SimulIDE) y código .ino(SimulIDE) ejecutados en el simulador gratuito SimulIDE









## **Conclusión**

El sistema logró detectar temperaturas mayores a 25 °C mediante un modelo de Machine Learning entrenado en TensorFlow y simulado con SimulIDE, integrando inteligencia artificial y electrónica de bajo costo (TinyML).

Este proyecto demuestra cómo un modelo ligero puede implementarse incluso en entornos simulados, permitiendo optimizar procesos de monitoreo en tiempo real.

### **Link de GitHub:**

<https://github.com/VJMontero777/Tarea09.git>

## **Actividades para el Estudiante**

### **1. ¿Qué papel juega OpenCV en la visualización de los datos de temperatura?**

OpenCV permite mostrar en pantalla las alertas visuales y mensajes en tiempo real cuando la temperatura cambia o supera el límite establecido.

### **2. ¿Qué diferencia existe entre el modelo original de TensorFlow y el convertido para TinyML?**

El modelo original de TensorFlow es pesado y se usa en PC, mientras que el de TinyML está optimizado para funcionar en microcontroladores con poca memoria.

### **3. ¿Qué acciones se toman cuando la temperatura sobrepasa los niveles de seguridad?**

Cuando la temperatura pasa los 25 °C, se activa el buzzer y el LED, mostrando en el monitor serie el mensaje de alerta.

### **4. ¿Cómo puede mejorar el modelo de predicción con más datos?**

Si se entrena con más muestras y variaciones de temperatura, el modelo aprende mejor y predice con más precisión los cambios reales.

### **5. ¿Cómo evalúas el desempeño de este sistema en un contexto real de almacenamiento?**

Funciona de forma rápida y confiable, permitiendo detectar a tiempo aumentos de temperatura que podrían dañar productos perecibles.