UNIVERSIDAD AUTÓNOMA DE OCCIDENTE

FACULTAD DE INGENIERÍA

INTELIGENCIA ARTIFICIAL – PERIODO ACADÉMICO 2022-01

Miniproyecto 3 - Aplicación para reconocimiento de audio usando Redes Neuronales Artificiales en una tarjeta de desarrollo como Tiny ML

Sebastian Grajales Jaramillo - 2185781

Manuela Cuasmayan Feria - 2186143

Ing. Jesús Alfonso López Sotelo

Universidad Autónoma de Occidente
Facultad de Ingeniería
Departamento de Operaciones y Sistemas

Inteligencia Artificial en Dispositivos Móviles y Embebidos
Santiago de Cali
2022

TABLA DE CONTENIDO

INTRODUCCIÓN 3

1. OBJETIVOS DEL PROYECTO 4

2. DESCRIPCIÓN DEL PROYECTO 4

3. DIAGRAMA DE BLOQUES Y CÓDIGO ARDUINO 6

4. HARDWARE UTILIZADO 8

5. RECOPILACIÓN DE DATOS 9

6. PLANTEAMIENTO DE LA SOLUCIÓN EN EDGE IMPULSE 10

7. PLANTEAMIENTO DE LA SOLUCIÓN EN TENSORFLOW-KERAS 15

8. RESULTADOS 22

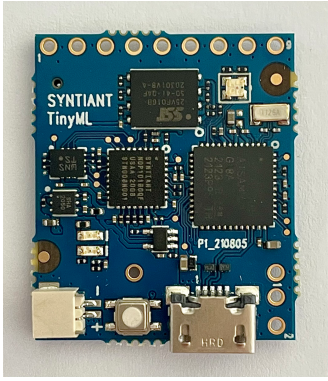
9. PROBLEMAS QUE SE PRESENTARON EN EL PROYECTO Y SUS SOLUCIONES 23

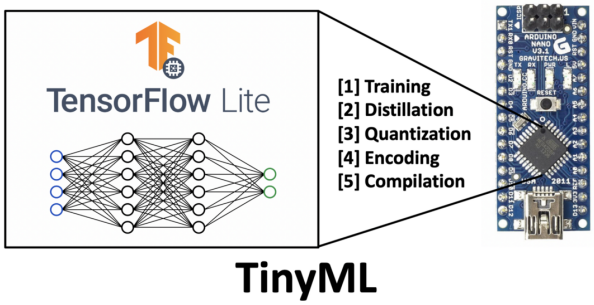
10. CONCLUSIONES 23

REFERENCIAS 24

INTRODUCCIÓN

Tiny ML es una plataforma de hardware y software de código abierto que permite a los desarrolladores implementar y ejecutar modelos de aprendizaje automático en dispositivos extremadamente pequeños y de bajo consumo. La plataforma se compone de una serie de herramientas y bibliotecas de código abierto, así como una serie de tarjetas de desarrollo de bajo costo y de bajo consumo. Entre las tarjetas de desarrollo se encuentra el TensorFlow Lite for Microcontrollers Development Board, que es una tarjeta de desarrollo de bajo costo y de bajo consumo que permite a los desarrolladores implementar y ejecutar modelos de aprendizaje automático en dispositivos extremadamente pequeños. La tarjeta de desarrollo está basada en el microcontrolador STM32F405RG y cuenta con 1 GB de memoria Flash y 128 MB de memoria RAM. La tarjeta de desarrollo también incluye una serie de periféricos, como un acelerómetro, un giroscopio, un sensor de temperatura, un sensor de humedad, un sensor de luz, un sensor de proximidad, un micrófono, un altavoz, una pantalla OLED, un puerto USB, una tarjeta SD, un conector para auriculares, un conector para una batería, y un botón de reset. La tarjeta de desarrollo es compatible con el sistema operativo Linux, así como con el IDE de Arduino.





1. OBJETIVOS DEL PROYECTO

- El objetivo de esta aplicación es reconocer el audio de una persona que habla y transformarlo en texto. Se usará una red neuronal artificial para el reconocimiento de audio. La red neuronal se entrenará usando datos de audio de mi compañera Manuela. Se usará una tarjeta de desarrollo para ejecutar la red neuronal. La tarjeta de desarrollo se encargará de la ejecución de la red neuronal y del procesamiento de audio.

- El objetivo de esta aplicación es reconocer el audio de una persona hablando y, a partir de este reconocimiento, activar o desactivar un dispositivo electrónico.

- Conexión entre 2 embebidos mediante el celular

2. DESCRIPCIÓN DEL PROYECTO

El proyecto consiste en construir una casa inteligente que utilice tecnología Tiny ML e inteligencia artificial para controlar todos los aspectos de la vida doméstica. La casa será capaz de aprender y adaptarse a las necesidades de sus ocupantes, maximizando su confort y eficiencia. Todos los dispositivos y sistemas de la casa estarán interconectados y serán controlados por una central de inteligencia artificial. Esto permitirá que la casa se adapte y cambie según las necesidades de sus ocupantes, haciendo la vida más fácil y agradable.

El proyecto se divide en tres fases principales:

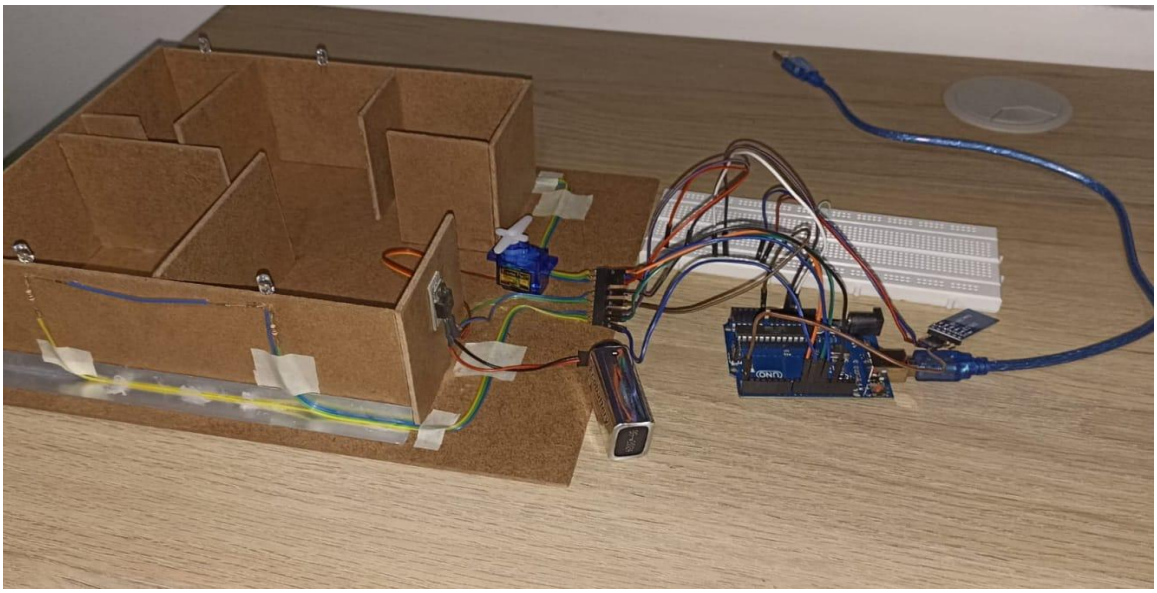
1) En la primera fase, se construirá la casa y se instalarán todos los dispositivos y sistemas necesarios.

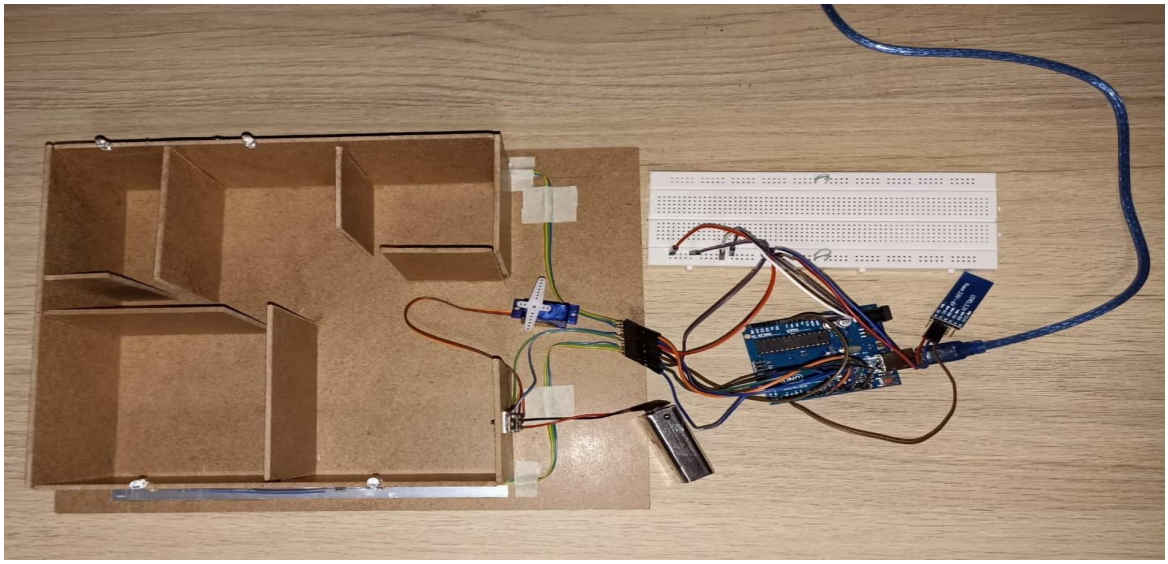
2) En la segunda fase, se entrenará a la inteligencia artificial para que controle todos los aspectos de la vida doméstica.

3) En la tercera y última fase, se pondrá la casa en funcionamiento y se evaluarán los resultados.

PRIMERA FASE

Fotos del proyecto en desarrollo:

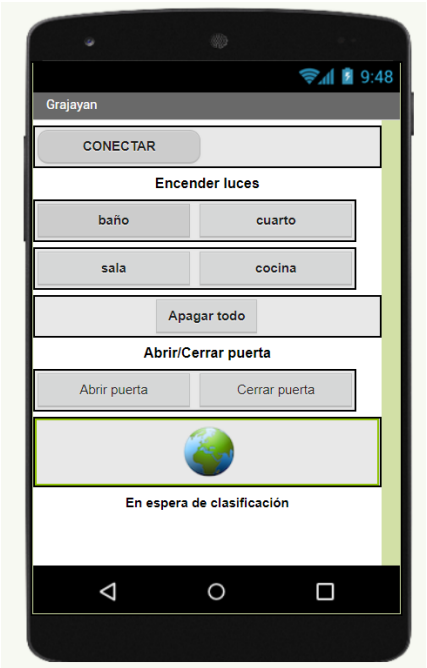




3. DIAGRAMA DE BLOQUES Y CÓDIGO ARDUINO

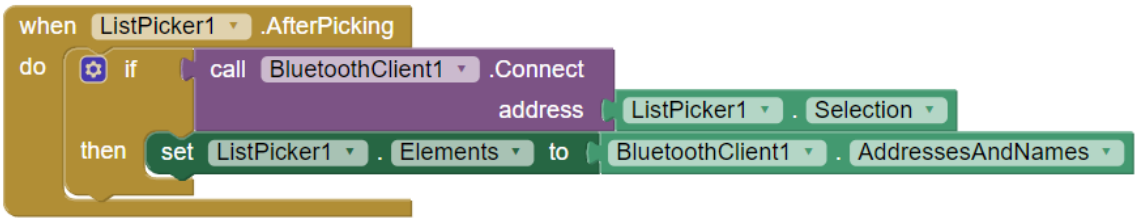
App inventor

Se diseñó la pantalla de la siguiente manera.

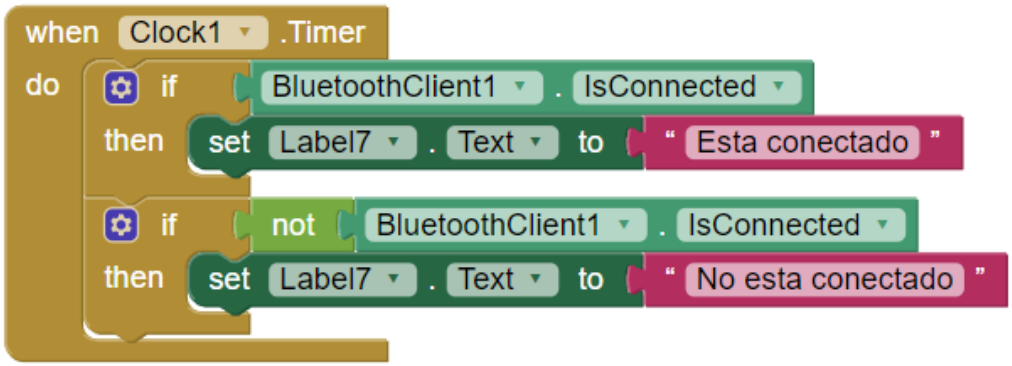


- Bloques
  Al presionar el botón de CONECTAR de la pantalla, se despliega la lista de dispositivos para establecer conexión bluetooth

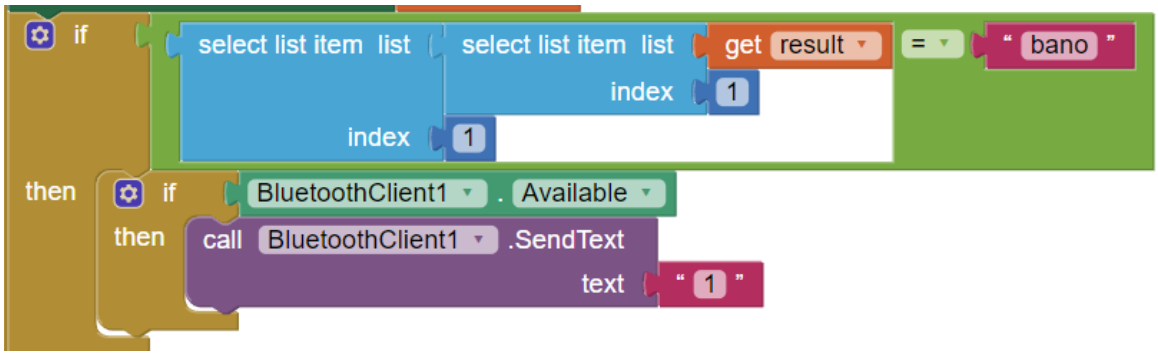
Se establece la conexión al seleccionar el dispositivo .



En la pantalla aparecerá un texto indicando si está conectado o no lo está.



Se clasifica el audio grabado, y si lo clasifica como la clase Bano, enviará por bluetooth el texto 1 al Arduino Uno, y lo mismo para las demás clases variando el texto a enviar.



- Código Arduino

Se definen las variables y sus puertos físicos del arduino.

```
#include <Servo.h>
Servo servomotor;

int ledbano = 9;
int ledcuarto = 10;
int ledsala = 11;
int ledcocina = 12;
char action;
```

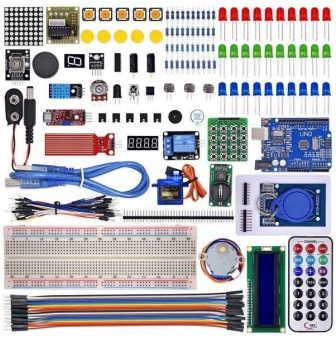
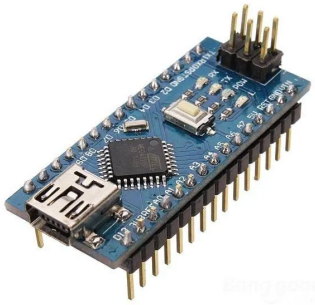
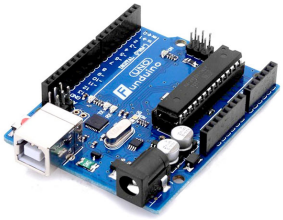
Se define el puerto físico del servomotor, y las pines de salida de los variables.

```
void setup() {
  servomotor.attach(A0);
   pinMode(ledcuarto, OUTPUT);
   pinMode(ledbano, OUTPUT);
   pinMode(ledsala, OUTPUT);
   pinMode(ledcocina, OUTPUT);
   Serial.begin(9600);

}
```

Si el Arduino recibe el carácter 1, encenderá el led correspondiente al del baño, lo mismo para las demás clases.

```
void loop() {

     if (action == '1')
  {
    digitalWrite(ledbano, HIGH);
  }
```

4. HARDWARE UTILIZADO

	NOMBRE	Descripción
	kit arduino Básico	Este kit dispone de todos los componentes básicos para usar un arduino.
	Arduino Nano 33	Este componente nos permitirá generar los datos para hacer el modelo y también comprobar la funcionalidad del modelo
	Arduino UNO	Este componente nos permitirá desarrollar el proyecto domótico para encender luces por medio de la voz.

	Servomotores	Nos permitirá mover las cerraduras de las puertas.
	Leds	Actuarán como una simulación de una lámpara encendida, que podremos controlar a través de la voz.

5. RECOPILACIÓN DE DATOS

SEGUNDA FASE

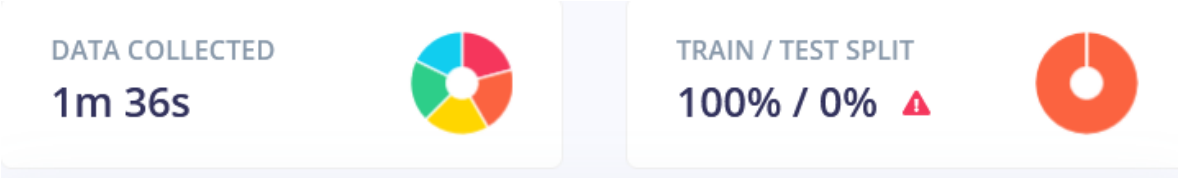
Para realizar el modelo se realizaron los siguientes pasos:

1. **Generar data set**

   Para esto utilizamos la plataforma Edge Impulse por medio del Arduino Nano 33 BLE para generar el data set. Se usaron 5 categorías con un registro de 1 segundo aproximadamente por cada uno:

   1. Baño
   2. Cocina
   3. Cuarto
   4. Puerta
   5. Sala

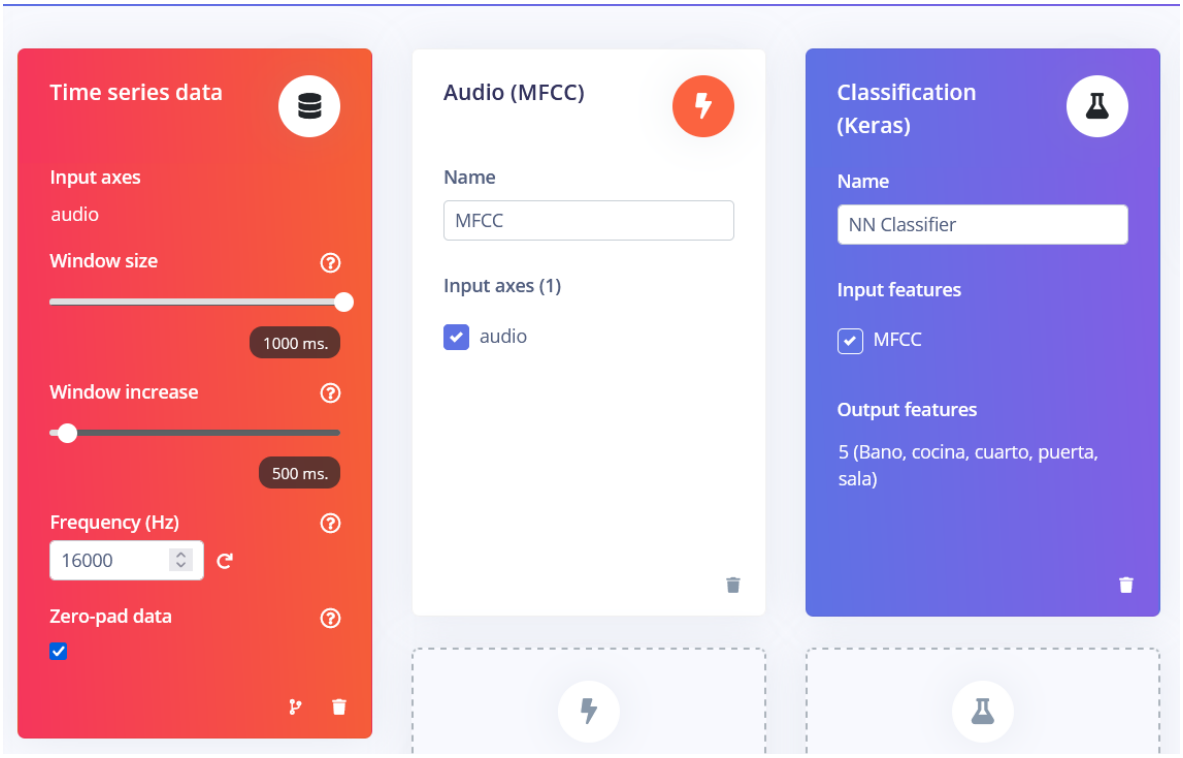
El total de datos recolectados fue de 1 min y 36, y el 100% de los datos son para entrenamiento



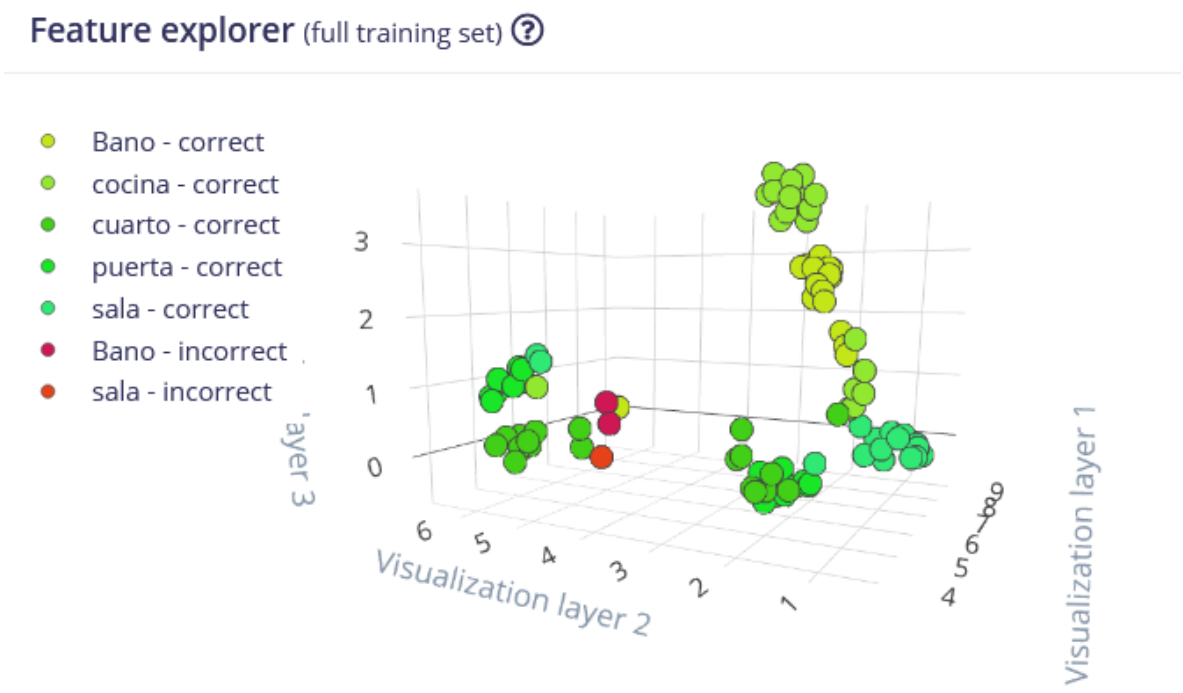
6. PLANTEAMIENTO DE LA SOLUCIÓN EN EDGE IMPULSE

Una vez recopilados los datos, se procede a diseñar el impulso que ofrece la plataforma Edge Impulse. Se utilizó el bloque de procesamiento de señales de audio “MFCC” para reconocimiento de voz humana, y también el bloque de aprendizaje de red neuronal.

Se estableció una ventana de tamaño 1000 ms y aumento de tamaño de 500 ms ya que es un modelo a base de sonido


Time series data
Input axes
audio
Window size
1000 ms.
Window increase
500 ms.
Frequency (Hz)
16000
Zero-pad data
Audio (MFCC)
Name
MFCC
Input axes (1)
audio
Classification (Keras)
Name
NN Classifier
Input features
MFCC
Output features
5 (Bano, cocina, cuarto, puerta, sala)

Características del modelo:



NN classifier:

Como se puede observar en la imagen, seleccionamos el balanceo automático de datos y le añadimos 2 capas de redes convolucionales 1D cada uno de 8 y 16 neuronas artificiales y un kernel de tamaño 3. También se le añadió un dropout y un Flatter layer para que el modelo se entrene bien.

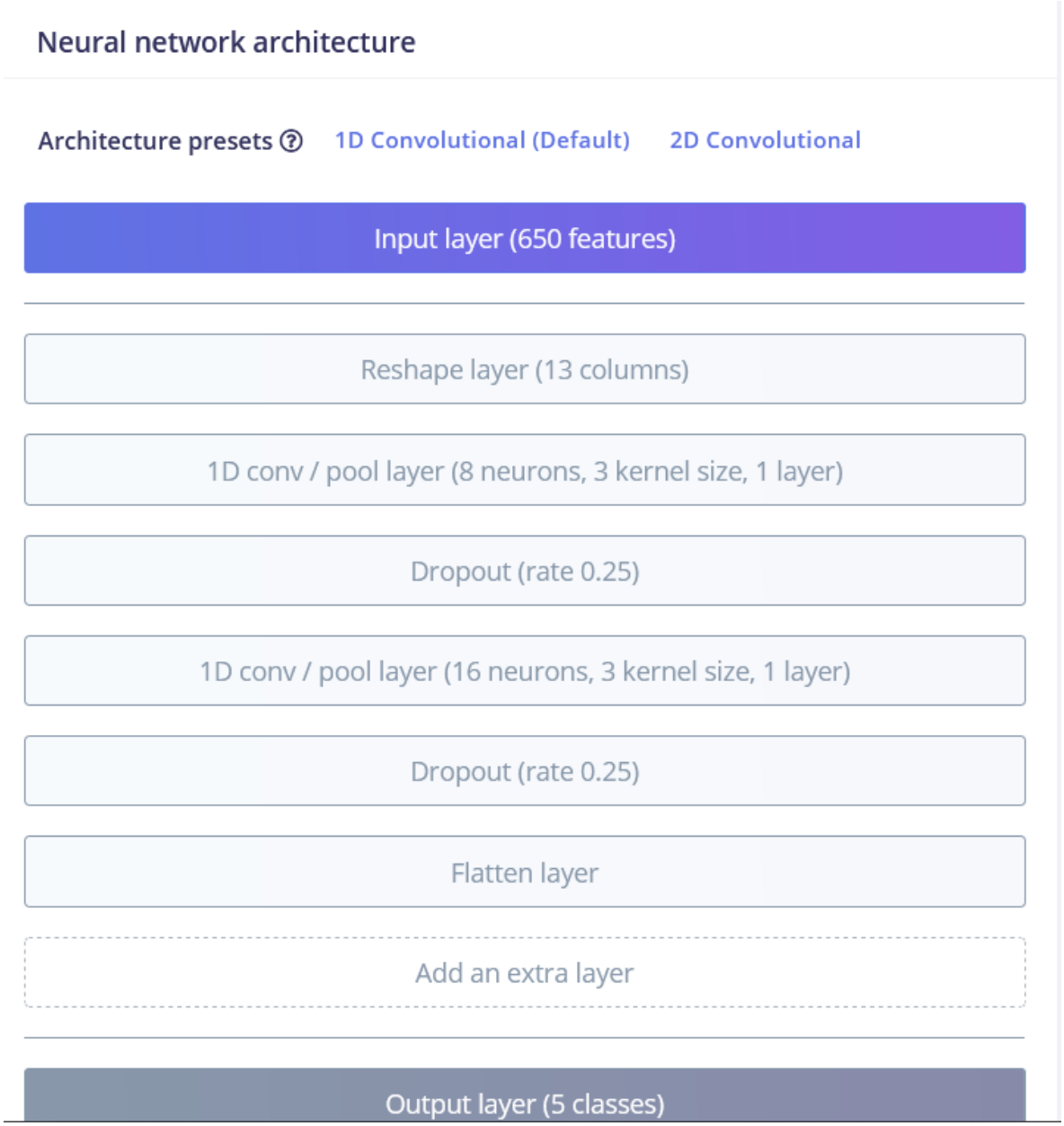
Neural Network settings

Training settings

Number of training cycles	100
Learning rate	0.005
Validation set size	20 %
Auto-balance dataset	☑

Audio training options

Data augmentation	☐


Neural network architecture
Architecture presets
1D Convolutional (Default)
2D Convolutional
Input layer (650 features)
Reshape layer (13 columns)
1D conv / pool layer (8 neurons, 3 kernel size, 1 layer)
Dropout (rate 0.25)
1D conv / pool layer (16 neurons, 3 kernel size, 1 layer)
Dropout (rate 0.25)
Flatten layer
Add an extra layer
Output layer (5 classes)

Matriz de confusión del modelo:

Last training performance (validation set)

ACCURACY
85.0%

LOSS
2.01

Confusion matrix (validation set)

	BANO	COCINA	CUARTO	PUERTA	SALA
BANO	66.7%	16.7%	0%	16.7%	0%
COCINA	0%	100%	0%	0%	0%
CUARTO	0%	0%	100%	0%	0%
PUERTA	0%	0%	0%	100%	0%
SALA	0%	0%	50%	0%	50%
F1 SCORE	0.80	0.80	0.93	0.86	0.67

7. PLANTEAMIENTO DE LA SOLUCIÓN EN TENSORFLOW-KERAS

Primero se descarga los datos, luego se extrae las características mediante un bloque de procesamiento para ya entrenar el modelo

Descargar los datos -después de extraer las características mediante un bloque de procesamiento- para poder entrenar un modelo de aprendizaje automático.

```python
import numpy as np
import requests

API_KEY = 'ei_d3a9d3ed55f124c19ae356f858bf2957fc232ef389cf0e4252874aa5bafc4c72'

def download_data(url):
    response = requests.get(url, headers={'x-api-key': API_KEY})
    if response.status_code == 200:
        return response.content
    else:
        print(response.content)
        raise ConnectionError('Could not download data file')

X = download_data('https://studio.edgeimpulse.com/v1/api/109095/training/5/x')
Y = download_data('https://studio.edgeimpulse.com/v1/api/109095/training/5/y')
```

Luego de haber extraído las características de los datos, se procede a almacenar los datos en un archivo temporal, y cargarlos de nuevo a través de Numpy.

Almacenar los datos en un archivo temporal, y cargarlos de nuevo a través de Numpy.

```python
with open('x_train.npy', 'wb') as file:
    file.write(X)
with open('y_train.npy', 'wb') as file:
    file.write(Y)
X = np.load('x_train.npy')
Y = np.load('y_train.npy')[:,0]
```

Luego se define las etiquetas para dividir los datos en un conjunto de pruebas y otro de entrenamiento

```python
import sys, os, random
import tensorflow as tf
from sklearn.model_selection import train_test_split

import logging
tf.get_logger().setLevel(logging.ERROR)
os.environ['TF_CPP_MIN_LOG_LEVEL'] = '2'

# Set random seeds for repeatable results
RANDOM_SEED = 3
random.seed(RANDOM_SEED)
np.random.seed(RANDOM_SEED)
tf.random.set_seed(RANDOM_SEED)

classes_values = [ "Bano", "cocina", "cuarto", "puerta", "sala" ]
classes = len(classes_values)

Y = tf.keras.utils.to_categorical(Y - 1, classes)

X_train, X_test, Y_train, Y_test = train_test_split(X, Y, test_size=0.2, random_state=1)

input_length = X_train[0].shape[0]

train_dataset = tf.data.Dataset.from_tensor_slices((X_train, Y_train))
validation_dataset = tf.data.Dataset.from_tensor_slices((X_test, Y_test))

def get_reshape_function(reshape_to):
    def reshape(image, label):
        return tf.reshape(image, reshape_to), label
    return reshape

callbacks = []

```

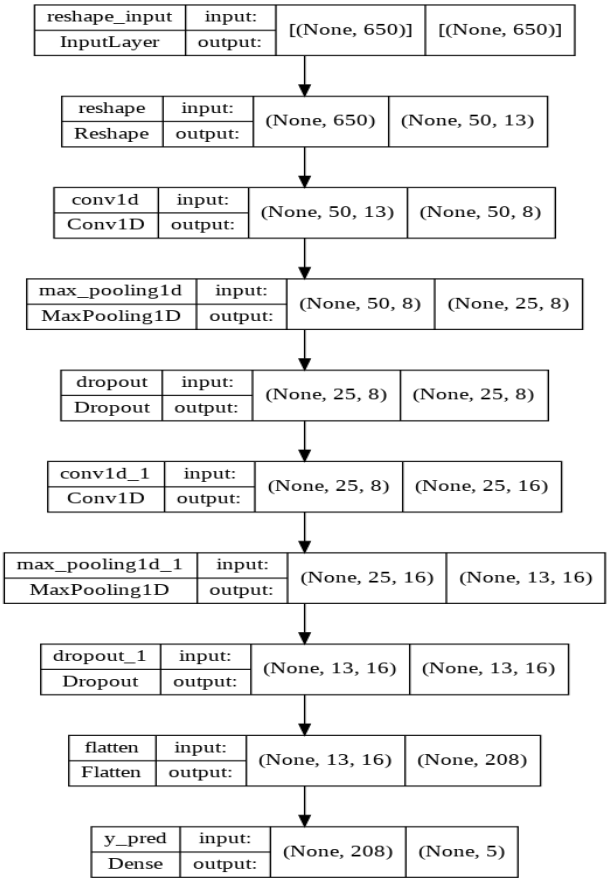
Luego de haber dividido los datos con etiquetas, se entrena el modelo

```python
import tensorflow as tf
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense, InputLayer, Dropout, Conv1D, Conv2D, Flatten, Reshape, MaxPooling1D, MaxPooling2D, BatchNormalization, TimeDistributed
from tensorflow.keras.optimizers import Adam
from keras.utils import np_utils
import matplotlib.pyplot as plt

# model architecture
model = Sequential()
model.add(Reshape((int(input_length / 13), 13), input_shape=(input_length, )))
model.add(Conv1D(8, kernel_size=3, activation='relu', padding='same'))
model.add(MaxPooling1D(pool_size=2, strides=2, padding='same'))
model.add(Dropout(0.25))
model.add(Conv1D(16, kernel_size=3, activation='relu', padding='same'))
model.add(MaxPooling1D(pool_size=2, strides=2, padding='same'))
model.add(Dropout(0.25))
model.add(Flatten())
model.add(Dense(classes, activation='softmax', name='y_pred'))

# this controls the learning rate
opt = Adam(learning_rate=0.005, beta_1=0.9, beta_2=0.999)
EPOCHS = 100
# this controls the batch size, or you can manipulate the tf.data.Dataset objects yourself
BATCH_SIZE = 32
train_dataset = train_dataset.batch(BATCH_SIZE, drop_remainder=False)
validation_dataset = validation_dataset.batch(BATCH_SIZE, drop_remainder=False)

# train the neural network
model.compile(loss='categorical_crossentropy', optimizer=opt, metrics=['accuracy'])
model.fit(train_dataset, epochs=EPOCHS, validation_data=validation_dataset, verbose=2, callbacks=callbacks)
```

Aquí se define la arquitectura del modelo

```
# model architecture
model = Sequential()
model.add(Reshape((int(input_length / 13), 13), input_shape=(input_length, )))
model.add(Conv1D(8, kernel_size=3, activation='relu', padding='same'))
model.add(MaxPooling1D(pool_size=2, strides=2, padding='same'))
model.add(Dropout(0.25))
model.add(Conv1D(16, kernel_size=3, activation='relu', padding='same'))
model.add(MaxPooling1D(pool_size=2, strides=2, padding='same'))
model.add(Dropout(0.25))
model.add(Flatten())
model.add(Dense(classes, activation='softmax', name='y_pred'))
```

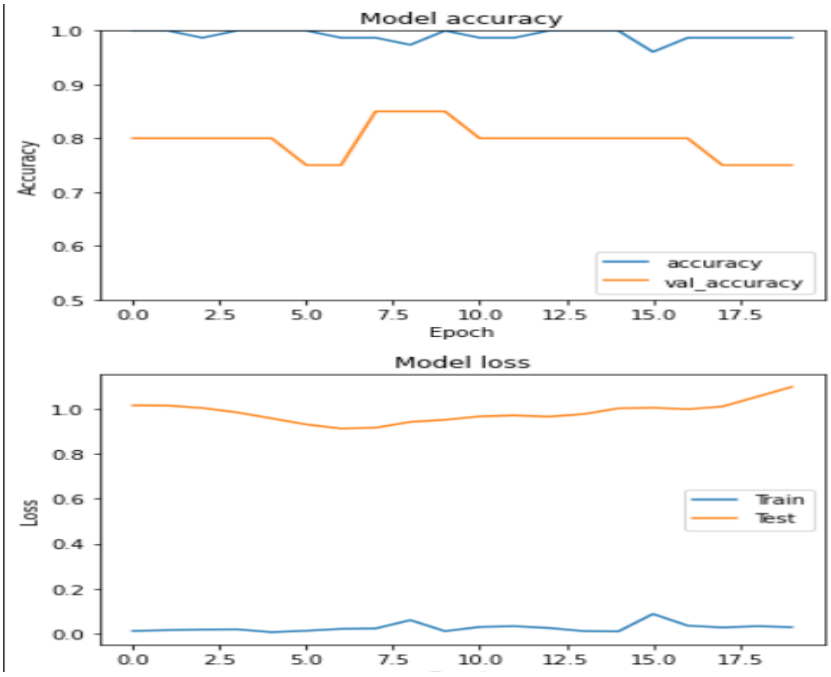
Como se puede observar en la imagen, es la gráfica del modelo



Luego se coloca el código con pl plot para graficar la precisión y pérdida de datos del modelo

Graficar evolución del modelo

```
plt.plot(history.history['accuracy'], label='accuracy')
plt.plot(history.history['val_accuracy'], label = 'val_accuracy')
plt.title('Model accuracy')
plt.xlabel('Epoch')
plt.ylabel('Accuracy')
plt.ylim([0.5, 1])
plt.legend(loc='lower right')
plt.show()

# Plot training & validation loss values
plt.plot(history.history['loss'])
plt.plot(history.history['val_loss'])
plt.title('Model loss')
plt.ylabel('Loss')
plt.xlabel('Epoch')
plt.legend(['Train', 'Test'], loc='best')
plt.show()
```



Luego se evalúa el modelo y como podrán ver, a cierta un 75%

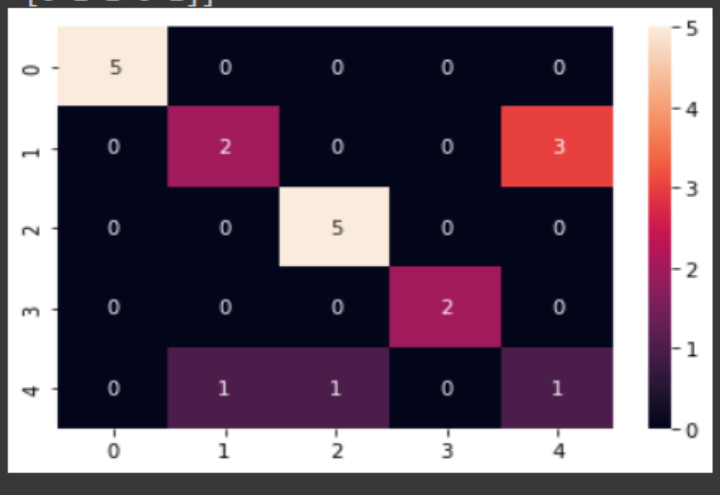
```
[ ]  1 # Evaluación del modelo
     2 Score = model.evaluate(X_test, Y_test, verbose=0)[1]
     3
     4 print(np.round(Score,decimals=4))
     5
     6 outputTrain=model.predict(X_test)

     0.75
```

Luego se realiza la matriz de confusión como podrán ver en ambas imágenes

Matriz de confusion del modelo

```
[ ]
from sklearn.metrics import classification_report, confusion_matrix
import seaborn as sns
#import matplotlib.pyplot as plt
import pandas as pd

Y_test_class = np.argmax(Y_test,axis=1)
Y_pred_class = np.argmax(outputTrain,axis=1)

# Cálculo de las métricas
print(classification_report(Y_test_class, Y_pred_class)) # Precision , Recall, F1-Score & Support
cm = confusion_matrix(Y_test_class, Y_pred_class)
print(cm)
# Visualización de la matriz de confusion en un mapa de calor
df_cm = pd.DataFrame(cm)
heatmap = sns.heatmap(df_cm, annot=True, fmt="d")
```

```
              precision    recall  f1-score   support

           0       1.00      1.00      1.00         5
           1       0.67      0.40      0.50         5
           2       0.83      1.00      0.91         5
           3       1.00      1.00      1.00         2
           4       0.25      0.33      0.29         3

    accuracy                           0.75        20
   macro avg       0.75      0.75      0.74        20
weighted avg       0.76      0.75      0.75        20

[[5 0 0 0 0]
 [0 2 0 0 3]
 [0 0 5 0 0]
 [0 0 0 2 0]
 [0 1 1 0 1]]
```



Una vez finalizado el entrenamiento del modelo y evaluación de la misma, se procede a guardar el modelo en formato Keras y Tensorflow

```
[ ]  1 # The model is saved in Keras format
     2 model.save('proyecto_final.h5')
```

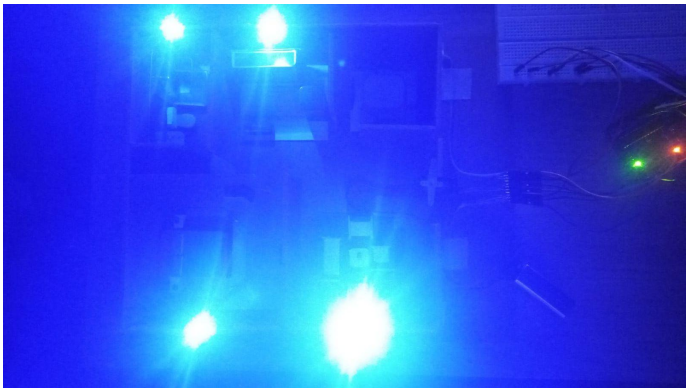
```
[ ]  1 # The model is saved in Tensorflow format
     2 model.save('proyecto_final')
```

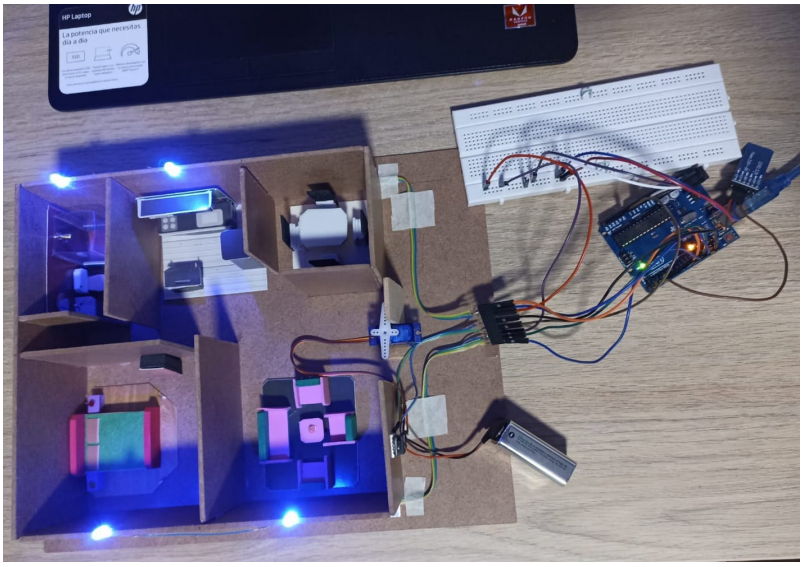
También se guardará el modelo en TensorFlow Lite para dispositivos embebidos de bajos recursos para que pueda ser ejecutado sin ningún problema

```
[ ]  # Model is converted to TensorFlow Lite
     converter = tf.lite.TFLiteConverter.from_keras_model(model)

[ ]  model_tflite = converter.convert()

     WARNING:absl:Buffer deduplication procedure will be skipped when flatbuffer library is not properly loaded

[ ]  # The model is saved
     open("/content/proyecto3.tflite","wb").write(model_tflite)

     12596

[ ]  # Convert a model with quantization
     converter = tf.lite.TFLiteConverter.from_keras_model(model)
     converter.optimizations = [tf.lite.Optimize.DEFAULT]
     tflite_quant_model = converter.convert()
```

8. RESULTADOS

TERCERA FASE





El proyecto domótico con tecnología Tiny ML e inteligencia artificial resultó en una gran mejora en el hogar. La tecnología Tiny ML se utilizó para controlar el hogar, y la inteligencia artificial se utilizó para apoyar el proyecto. El proyecto mejoró la eficiencia del hogar, la seguridad del hogar y la comodidad del hogar. La tecnología Tiny ML se encargó de la iluminación y también se encargó de la seguridad del hogar, ya que se conectó con un servomotor para abrir y cerrar la puerta principal del hogar.

LINK DEL PROYECTO EN EDGE IMPULSE:

https://studio.edgeimpulse.com/public/109095/latest

9. PROBLEMAS QUE SE PRESENTARON EN EL PROYECTO Y SUS SOLUCIONES

- Hubo complicaciones para que el computador encontrará el puerto del Arduino Nano 33 BLE, se debía estar constantemente revisando que el puerto correspondiera con el del Arduino.

- Hubo complicaciones para hacer la matriz de confusión porque tomaba los datos de entrenamiento, se debió cambiar unas variables del código.

- Fue necesario optimizar el modelo en TensorFlow Lite para que el arduino Nano 33 pudiera ejecutar el modelo.

10. CONCLUSIONES

Las conclusiones del miniproyecto son las siguientes:

1. La tecnología Tiny ML y la inteligencia artificial pueden simplificar y mejorar significativamente la eficiencia y la funcionalidad de un proyecto domótico.

2. La tecnología Tiny ML y la inteligencia artificial pueden hacer que un proyecto domótico sea más seguro y confiable, al tiempo que simplifica el uso y la gestión de los dispositivos y sistemas domóticos.

3. La tecnología Tiny ML y la inteligencia artificial pueden permitir que un proyecto domótico sea más eficiente en el uso de recursos, lo que puede resultar en un ahorro significativo de costos.

4. La tecnología Tiny ML y la inteligencia artificial pueden simplificar el proceso de instalación y configuración de un proyecto domótico, lo que puede reducir significativamente el tiempo y el esfuerzo requeridos.

5. La tecnología Tiny ML y la inteligencia artificial pueden mejorar la capacidad de un proyecto domótico para detectar y responder a eventos y condiciones anómalas, lo que puede ayudar a prevenir problemas y asegurar un funcionamiento óptimo.

REFERENCIAS

Baez Alcocer, C. R., Cántaro Hernández, M. A., & Cueva Mijahuanca, R. A. (2016). Diseño e implementación de un Sistema Domótico para una vivienda estándar.

DE, M. L. C. D. V. (2015). IMPLEMENTACIÓN DE INTELIGENCIA ARTIFICIAL (IA) EN DOMÓTICA.

Espinoza Saavedra, M. T., & López Hurtado, C. A. (2015). Desarrollo de soluciones con domótica.

López Aguilar, E., & Garrido Rubio, A. Los sistemas domóticos controlados por voz. El caso de los asistentes inteligentes.

Cabezas García, W., & Jaime Neira, F. A. (2016). Diseño de un control domótico basado en una plataforma Open Source para viviendas.

Henríquez, M. R., & Palma, P. A. (2011). Control automático de condiciones ambientales en domótica usando redes neuronales artificiales. *Información tecnológica*, *22*(3), 125-139.