CENTRO DE CIENCIAS BÁSICAS

DEPARTAMENTO DE SISTEMÁS ELECTRÓNICOS

INGENIERÍA EN ELECTRÓNICA

APRENDIZAJE AUTOMÁTICO EN SISTEMAS EMBEBIDOS

DNN Detector de Letras

ALUMNA:

HIDALGO RODRÍGUEZ MELISSA

INTRODUCCIÓN

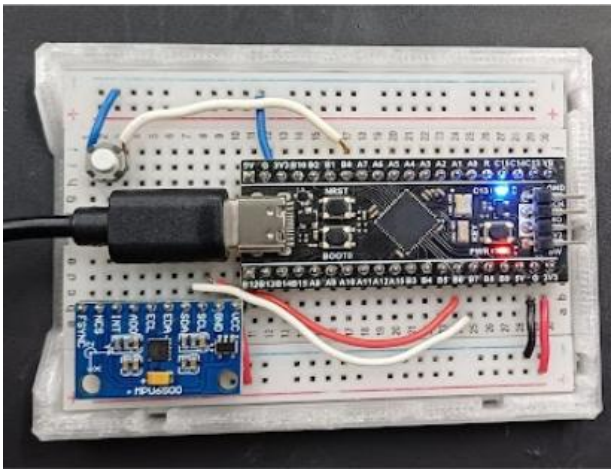
El presente reporte aborda un sistema distribuido entre un microcontrolador y Python para clasificar letras según el movimiento realizado en un IMU, en este caso las letras M, H y R, el modelo de IA utilizado fue una red neuronal densa (DNN). Para la implementación en hardware se hizo uso de una Black Pill STM32F401, un MPU6500, el puerto USB, el LED y el botón integrado de la misma tarjeta de desarrollo.

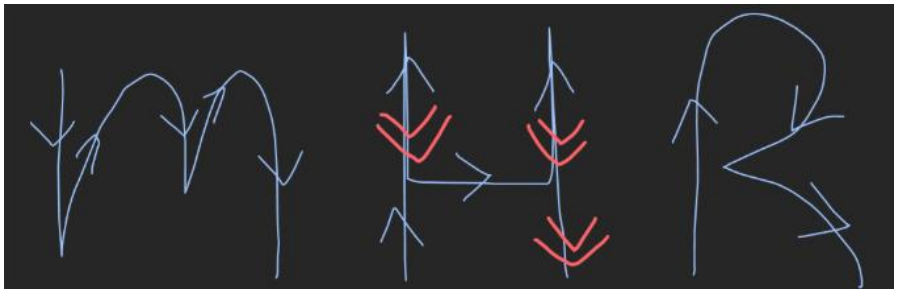
Los objetivos de la práctica son:

- Entrenar un modelo con TensorFlow e implementar la inferencia en un microcontrolador usando STM32 Cube AI.
- Laboratorio basado en proyectos: Desarrollar la aplicación de reconocimiento de actividad mediante perceptrón multicapa (MLP).
- Evaluar con y sin extracción de características. Ejecutar y evaluar el rendimiento en un microcontrolador STM32.

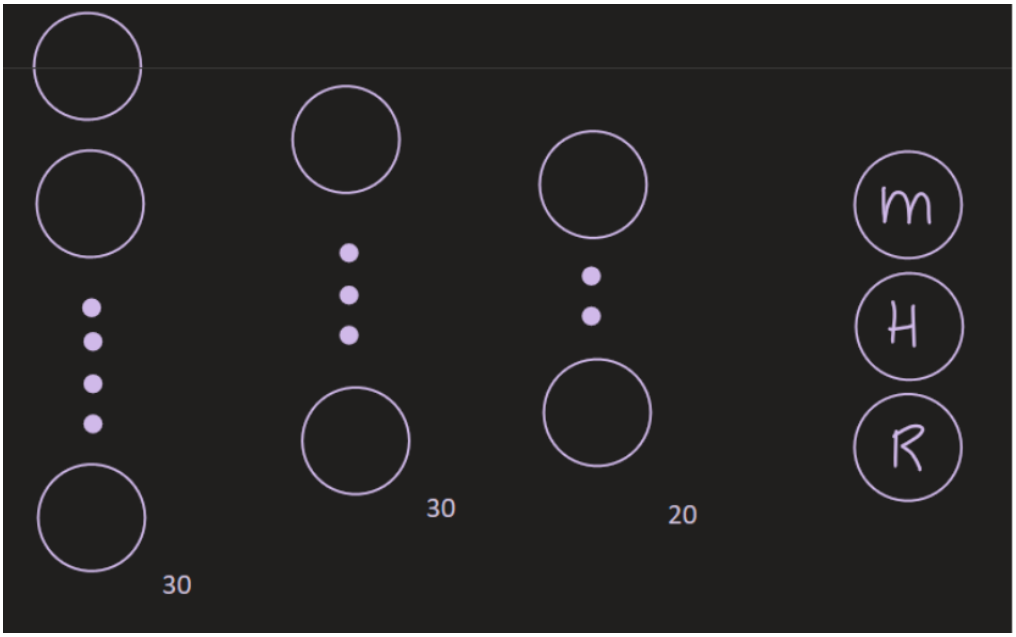
DESARROLLO

Después de la obtención de los datos de aceleración del MPU en el microcontrolador, se transfirieron a Python a través de un puerto COM y se guardaron en archivos csv, el dataset cuenta con 90 muestras de cada letra. Estos datos se procesaron en jupyter con el framework de TensorFlow para definir la arquitecutra de la red neuronal y realizar el entrenamiento. En las siguientes imagenes se muestra el movimiento que debe seguirse para cada letra, donde la salida USB del BlackPill debe estar apuntando a la izquierda.





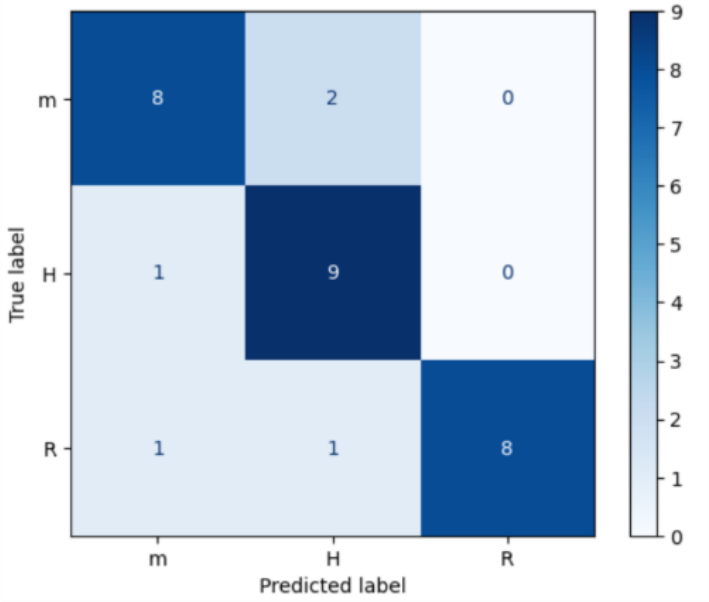
De manera gráfica la arquitectura de la red neuronal se muestra en la siguiente imagen.



MÉTRICAS Y MATRICES DE CONFUSIÓN

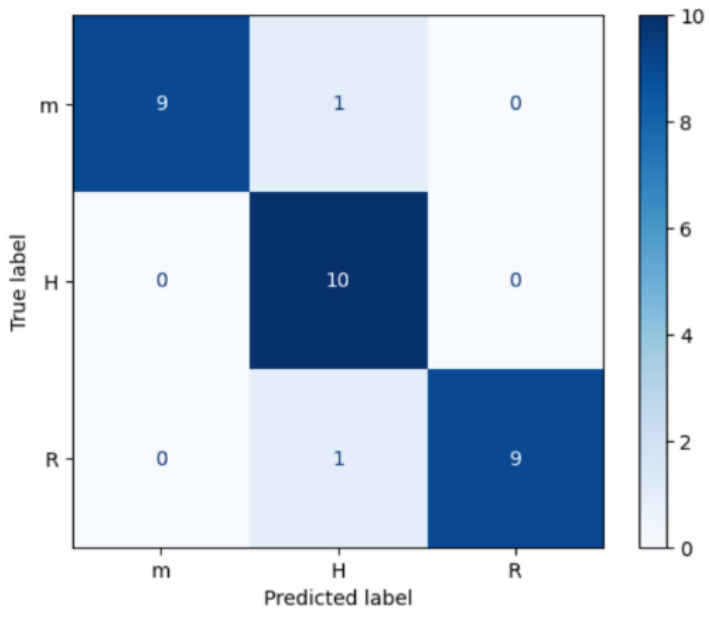
Las matrices de confusión y las métricas de las inferencias del modelo con reducción de dimensionalidad extrayendo las características en el jupyter son:

Exactitud	0.8333		
	m	H	R
Precisión	0.8	0.75	1
Recall	0.8	0.9	0.8
F1 score	0.8	0.81	0.88



Las matrices de confusión y las métricas de las inferencias del modelo con reducción de dimensionalidad extrayendo las características en el MCU son:

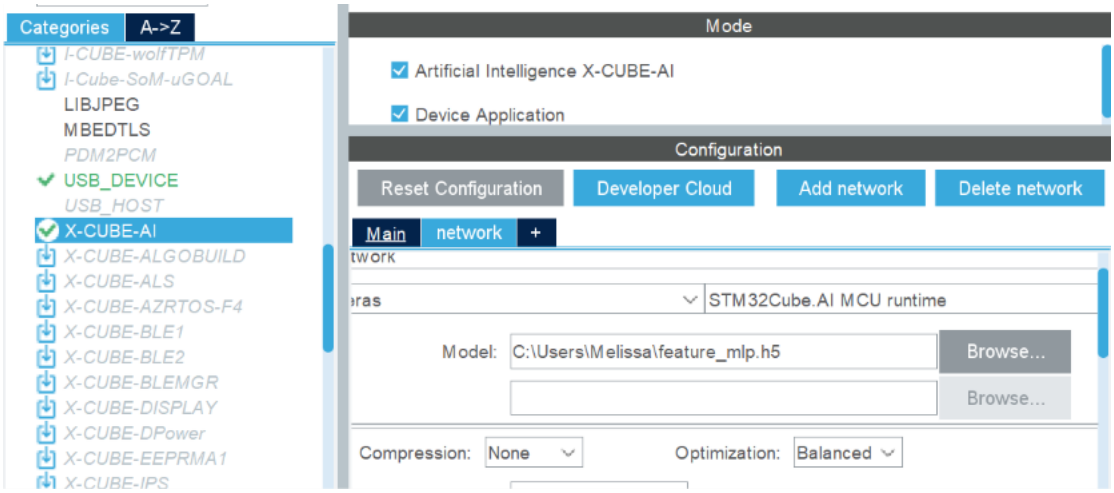
Exactitud	0.933		
	m	H	R
Precisión	1	0.83	1
Recall	0.9	1	0.9
F1 score	0.94	0.9	0.94



DISCUSIÓN

Para cargar el modelo al MCU se descargó la librería.

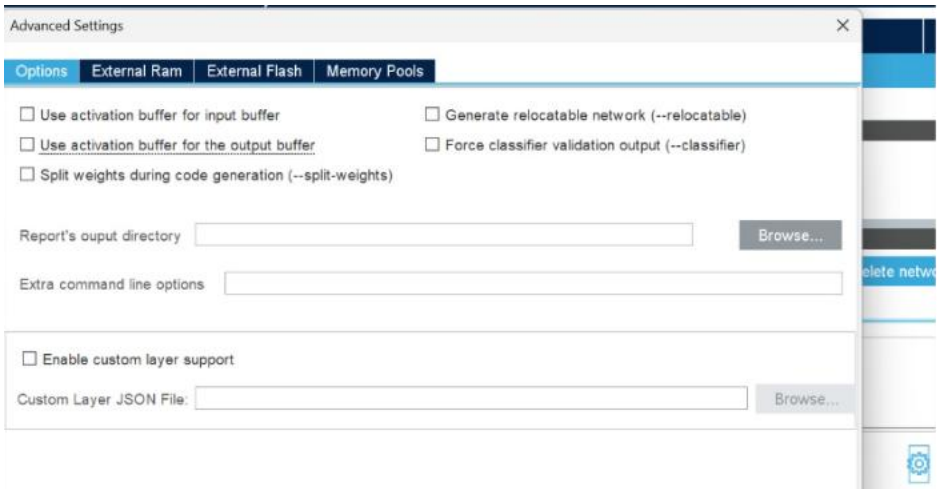
Se cargó el archivo .h5



Al analizar el modelo se tenían errores con la ruta, para extender el tamaño permitido de la ruta en cmd en la dirección donde se ubicaba el proyecto se ejecutó el siguiente comando.

```
reg add
"HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Control\FileSystem" /v
LongPathsEnabled /t REG_DWORD /d 1 /f
```

Para facilitar el manejo de los datos de entrada y salida se desactivaron las opciones de uso compartido del buffer. Por lo que integraron buffers de entrada y de salida diferentes.



Una vez completado el código de captura datos y despliegue de resultados mostrados en la sección anterior, se agregó una librería con la definición de printf para transmitir los datos por USB como puerto serial virtual y se agregaron las declaraciones de otras funciones que pedía el código, de lo contrario lanzaba error.

Una vez obtenidos los datos del MPU, para extraer los features se incluyó la librería StatisticsFunctions como muestra el siguiente manual.

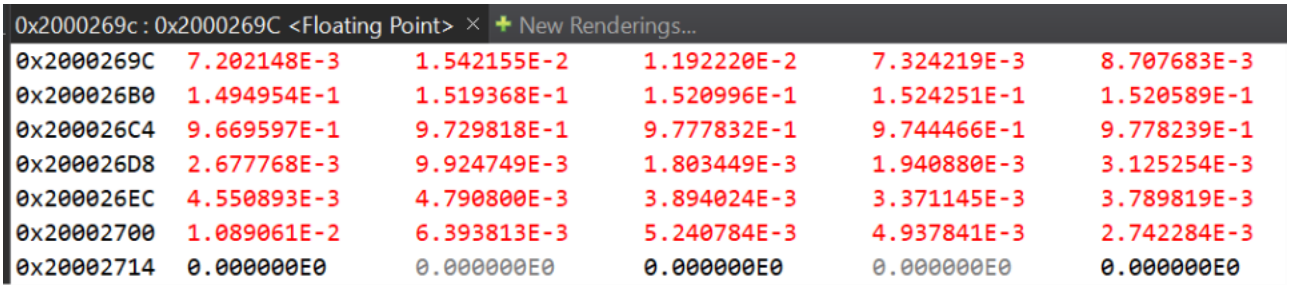
https://community.st.com/t5/stm32-mcus/how-to-integrate-cmsis-dsp-libraries-on-a-stm32-project/ta-p/666790

Para calcular las medias y desviaciones estándar, estas se capturaron en vectores individuales y en una matriz se acomodaron en le orden esperado por el modelo con la instrucción memcpy

```
memcpy(features_vector,      x_mean, NUM_WINDOWS * sizeof(float));
memcpy(features_vector + 5,  y_mean, NUM_WINDOWS * sizeof(float));
memcpy(features_vector + 10, z_mean, NUM_WINDOWS * sizeof(float));
memcpy(features_vector + 15, x_std,  NUM_WINDOWS * sizeof(float));
memcpy(features_vector + 20, y_std,  NUM_WINDOWS * sizeof(float));
memcpy(features_vector + 25, z_std,  NUM_WINDOWS * sizeof(float));
```

Para facilitar el cast y la lectura de la dirección se referencio un puntero al vector donde el modelo esperaba los datos y se copió la matriz de datos ordenados en esa dirección.

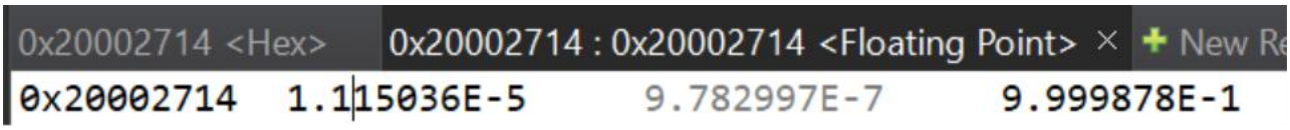
El llenado de datos se muestra en la siguiente imagen.



0x2000269c : 0x2000269C <Floating Point> × + New Renderings...

0x2000269C	7.202148E-3	1.542155E-2	1.192220E-2	7.324219E-3	8.707683E-3
0x200026B0	1.494954E-1	1.519368E-1	1.520996E-1	1.524251E-1	1.520589E-1
0x200026C4	9.669597E-1	9.729818E-1	9.777832E-1	9.744466E-1	9.778239E-1
0x200026D8	2.677768E-3	9.924749E-3	1.803449E-3	1.940880E-3	3.125254E-3
0x200026EC	4.550893E-3	4.790800E-3	3.894024E-3	3.371145E-3	3.789819E-3
0x20002700	1.089061E-2	6.393813E-3	5.240784E-3	4.937841E-3	2.742284E-3
0x20002714	0.000000E0	0.000000E0	0.000000E0	0.000000E0	0.000000E0

De manera muy similar el código indica la dirección de memoria donde se almacena la salida del modelo.



0x20002714 <Hex> 0x20002714 : 0x20002714 <Floating Point> × + New Re

0x20002714	1.115036E-5	9.782997E-7	9.999878E-1

CONCLUSIÓN

El experimento demostró que el uso de una Red Neuronal Densa (DNN) con extracción de características (media y desviación estándar) ofrece un buen rendimiento para la clasificación de movimientos ('m', 'H', 'R') en el sistema embebido. El modelo final implementado en el microcontrolador, utilizando la herramienta STM32 Cube AI, alcanzó una exactitud del 93.3%.

Además, de este experimento se concluye lo siguiente:

- La cantidad de datos de entrenamiento es crucial para hacer un modelo robusto y esta cantidad dependerá de cada situación, por ejemplo, se estima que si la similitud entre las letras hubiera sido menor el modelo hubiera sido eficiente con tal vez 60 muestras por cada letra.

- La reducción de características no siempre mejorará la calidad del modelo, es un fenómeno que depende de cada dataset y tipo de modelo a usar, sin embargo, reduce la dimensionalidad del modelo, aspecto esencial para los modelos de IA embebidos.

- Una exactitud de 1 en un modelo puede indicar que algo está mal, si en la exactitud del entrenamiento y de la evaluación se obtiene 1, es primordial hacer inferencias con datos nuevos para validar que el modelo es bueno.