CapibaraBot: An Educational Robotic System for Children Based on Playful Interaction and Accessible Hardware

1st Nelson Casa
National Polytechnic School
Computer Science Student
Quito, Ecuador
nelson.casa@epn.edu.ec

2nd Cristian Díaz
National Polytechnic School
Computer Science Student
Quito, Ecuador
cristian.diaz02@epn.edu.ec

3rd Wilson Inga
National Polytechnic School
Computer Science Student
Quito, Ecuador
wilson.inga@epn.edu.ec

4th Jessica Morales
National Polytechnic School
Computer Science Student
Quito, Ecuador
jessica.morales02@epn.edu.ec

5th Anthony Reinoso
National Polytechnic School
Computer Science Student
Quito, Ecuador
anthony.reinoso@epn.edu.ec

***Abstract*—This paper presents a low-cost educational robotic system aimed at children aged 2 to 5 years. It integrates a physical robot designed as a capybara, built with MDF wood and controlled via ESP32 and L298N, with a mobile video game application. The interaction between the physical robot and the virtual environment introduces early concepts of robotics, logic, and motor coordination. This project seeks to offer a playful and affordable alternative to existing commercial educational kits, tailored for educational contexts in Latin America.**

***Keywords*—Educational robotics, children, ESP32, video game, MDF prototyping, low-cost systems.**

I. Introduction

The rapid growth of technology and its increasing integration into education demand the development of tools that introduce young children to computational thinking and robotics. In this context, we present a tangible and playful solution designed for children aged 2 to 5: a robotic system called **CapibaraBot**, consisting of a physical robot shaped like a capybara built with MDF and controlled by an **ESP32 microcontroller and an L298N motor driver**, integrated with a **mobile video game application**.

The interaction between the physical robot and the virtual environment aims to stimulate cognitive development, motor coordination, and early understanding of technological concepts, especially in Latin American educational contexts.

A. Specific Objective

The main objective of this project is to develop a **low-cost educational robotic system** that includes a physical robot and a gamified mobile application. This system allows young children to intuitively explore basic concepts of movement, logic, and control. The project aims to:

- Spark technological curiosity through immersive and meaningful experiences.
- Promote psychomotor skills such as hand-eye coordination using a simple button interface (left, right, shoot).
- Provide an inclusive and economical educational tool as an alternative to expensive commercial kits.
- Demonstrate the connection between hardware and software through direct interaction with the robot.

B. Identified Problems

Although educational robotics is a powerful tool to foster logical thinking and creativity, its application in preschool settings faces several challenges:

- **Limited accessibility:** Most existing kits (e.g., LEGO Mindstorms, Bee-Bots) are prohibitively expensive for schools or families with low income.
- **Age-inappropriate tools:** There is a lack of solutions tailored specifically for children under five years old.
- **Hardware-software disconnect:** Many solutions focus only on software logic without including tangible interaction.
- **Low digital literacy:** Parents and teachers often lack the knowledge to integrate technological tools in early childhood education.

C. Problem–Solution Approach

To address the aforementioned problems, we propose an integrated and accessible system consisting of:

1) **A friendly-looking physical robot:** The robot is shaped like a capybara, built from laser-cut MDF, and controlled by ESP32 and L298N components.
2) **A simple and intuitive mobile game:** Players move the capybara using left/right/shoot buttons to dodge falling bombs and shoot an airplane.
3) **A modular and open architecture:** Built with locally available components to ensure replicability in Ecuadorian schools and homes.
4) **A playful pedagogical model:** Based on learning through hands-on play, combining real and virtual elements.

D. Contribution and Methodology

The project was carried out using a user-centered design methodology, including the following stages: requirements analysis, design, prototyping, and evaluation.

- **Requirements analysis:** Using *Hierarchical Task Analysis (HTA)* and persona modeling, we identified needs and constraints of children, parents, and educators. Emphasis was placed on simplicity, visual appeal, and safety.
- **Design alternatives:** We evaluated commercial educational robots such as LEGO Mindstorms and mBots. Due to their high cost (over $900 in some cases) and complexity, we opted for a custom solution using ESP32, motors, and MDF, making the system more appropriate for the local context.
- **Prototyping:**
  - **Low-fidelity:** Initial sketches and interface wireframes were created using Figma.
  - **High-fidelity:** The capybara model was designed in AutoCAD and physically built with laser-cut MDF. The mobile app includes a user interface with direction and shooting controls that mimic real robot movement.
- **Implementation and evaluation:** After assembling the robot and developing the app, we conducted usability tests with actual users (children and parents). Evaluation tools included the **System Usability Scale (SUS)** and direct observation based on **Nielsen's heuristics**. Results indicated high engagement, intuitive interaction, and clear educational potential.

II. Related Work

The CapibaraBot project combined the physical implementation of an educational toy shaped like a capybara with a mobile app designed to simulate a playful and educational gaming environment. The robot was built with laser-cut MDF and assembled on a sturdy, child-friendly chassis, adapted for use by children aged 2 to 5. Inside, the system uses an ESP32 microcontroller, along with an L298N motor controller module and DC motors, allowing the capybara to physically move left or right in response to commands sent from the app.

This app, developed with MIT App Inventor, depicts the capybara in a visual narrative where it must evade bombs dropped by an enemy plane flying overhead [5]. To introduce an adaptive and dynamic component, an artificial intelligence-based difficulty system was incorporated, fed by a set of simulated data that takes into account variables such as the number of bombs that hit the player, the number of projectiles intercepted, and the total duration of the game session. [1]

Using this dataset of more than 30,000 records, a Random Forest Regressor model was trained to predict the speed of the aircraft and the frequency with which it drops bombs, thus adjusting the level of difficulty in a personalized way according to the user's performance [2]. This model was evaluated using metrics such as mean absolute error (MAE), obtaining low values for both speed and bomb frequency predictions, which validates its effectiveness in responding to real-world scenarios.

Finally, this predictive engine was integrated with a graphical interface using input widgets, allowing the system's response to different game values to be tested without the need for multiple physical tests [3]. The fusion of hardware, software, artificial intelligence, and usability resulted in a meaningful, adaptive, and accessible learning environment that seeks to foster early skills in logic, coordination, and technology in Latin American educational contexts [4].

A. Circuit

The CapibaraBot circuit is based on an ESP32 microcontroller, which serves as the main component responsible for controlling the robot. This microcontroller is connected to an L298N motor driver module, which controls two DC motors that manage the robot's movement. The ESP32 receives control signals via Bluetooth from the mobile application, processes these signals, and drives the motors through the L298N. The microcontroller also manages the power connections for the system, which is powered by a 7.4V battery, supplying energy to both the ESP32 and the motors.

The circuit includes several key connections: the ESP32 pins control the direction and enable the motors through the L298N. Additionally, to stabilize the power supply and prevent voltage spikes, it is recommended to include capacitors in the power lines. This circuit design enables efficient communication between the hardware and the mobile application, ensuring that the robot moves correctly in response to the commands sent by the user.

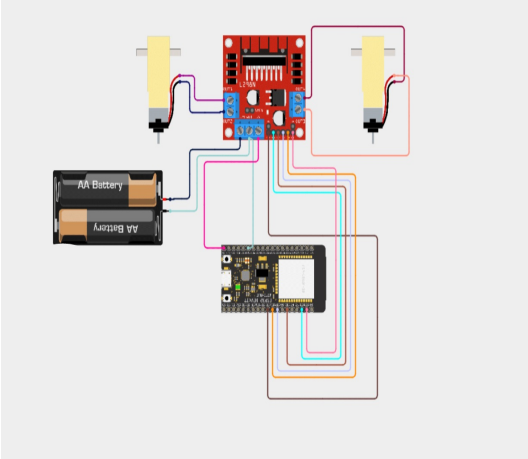

Fig. 1: Circuit

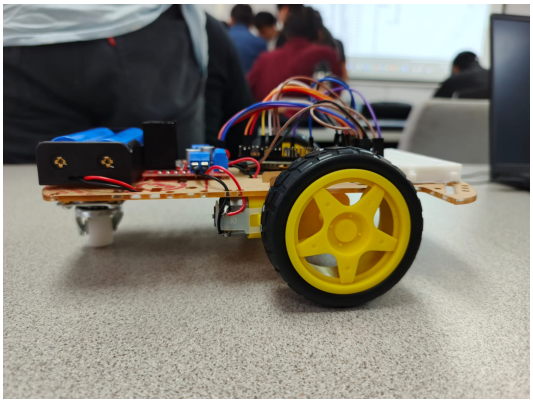

Fig. 2: Internal circuit construction

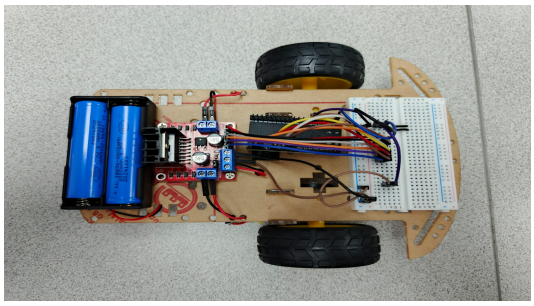

Fig. 3: Complete circuit model

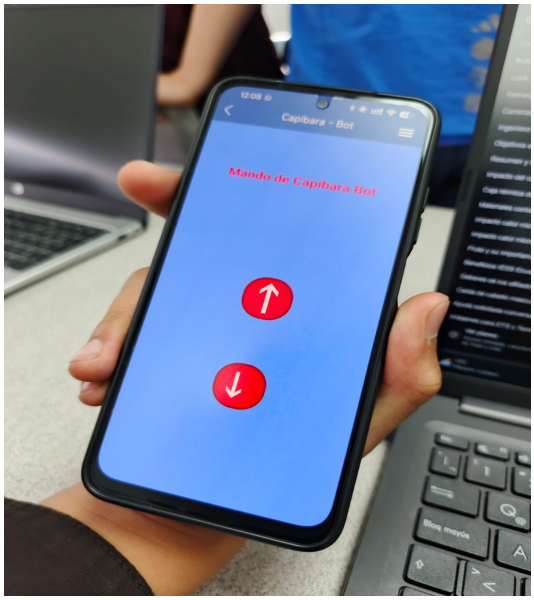

Fig. 4: Circuit manager via interface

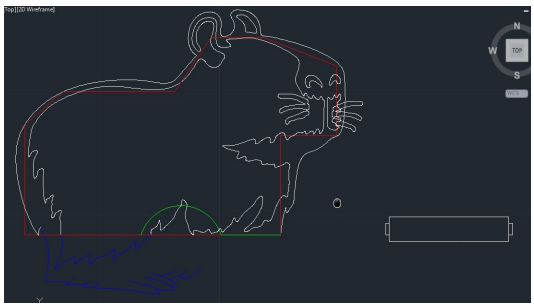

Fig. 5: interactive model design

Fig. 6: complete interactive project

III. System Evaluation

As part of the CapibaraBot system, a prediction model was developed that focuses on dynamically adjusting the game's difficulty based on user performance. This prediction takes into account key parameters such as the speed of the plane (the game's main enemy) and the frequency with which bombs or projectiles are launched, with the aim of maintaining an adaptive level of challenge appropriate to the cognitive and motor skills of children between the ages of 2 and 5.

During the evaluation of the model, mean absolute error (MAE) metrics were used to estimate the accuracy of the predictions generated. The total average MAE reached a value of 0.84, indicating acceptable performance for a low-complexity gamified educational environment. Specifically, an MAE of 0.76 was recorded in the prediction of aircraft speed, demonstrating a good level of accuracy in estimating the main dynamics of the game. On the other hand, the MAE in predicting the frequency of bombs was 0.92, showing a slightly higher deviation, possibly due to the variability of player behavior in short sessions.

In one of the prediction tests, simulated data was entered that resulted in an aircraft speed of level 4 (where 2 represents easy and 10 represents difficult) and a bomb frequency of level 7 (where 10 represents easy and 1 represents difficult). This result suggests that the model is capable of adapting the difficulty of the environment based on the user's pace, prioritizing a challenging but not frustrating experience.

A. Aircraft speed

Scatter plot showing the relationship between the actual and predicted values of the aircraft's speed. The blue points represent the model's predictions, while the dotted line indicates the perfect prediction. A strong positive correlation is observed, with most points closely aligned with the diagonal, indicating good model performance for this variable.

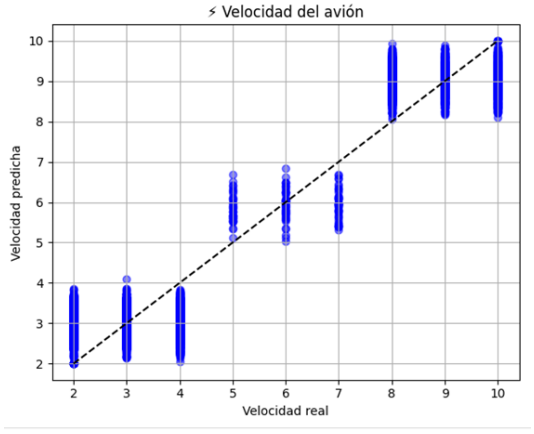

Fig. 7: Relationship between the actual and predicted speed of the aircraft. The model shows high accuracy, with points aligned diagonally

B. Pump frequency

Scatter plot showing the relationship between actual and predicted values for the frequency of bombs dropped. The orange dots correspond to the model's predictions. Although

there is a general upward trend, there is greater dispersion from the ideal line, especially at low values, suggesting that the model is less accurate for this variable.

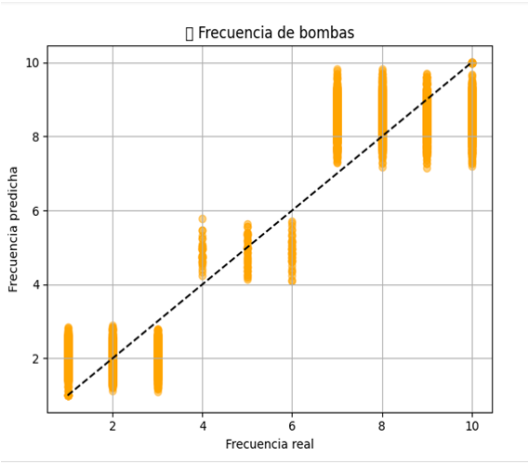

Fig. 8: Relationship between actual and predicted pump frequency. The model shows greater dispersion, indicating lower accuracy than in speed.

Both graphs show that the model performs well in predicting both aircraft speed and bomb frequency. However, the accuracy of speed predictions appears to be slightly higher than that of frequency, as the blue points are closer to the ideal line than the orange points. The discrepancies observed in the lower regions may indicate that the model needs adjustments or further training to improve estimation in those specific ranges. Overall, the model's performance is consistent and useful for predictive tasks in this context.

IV. User Evaluation

To evaluate the user experience with the CapibaraBot system, a validation session was conducted that included direct interaction with the functional prototype of the game. The participating user received a comprehensive explanation of the project's architecture, covering both physical and digital components.

Details were provided on the assembly of the physical toy based on the ESP32 microcontroller, its Bluetooth connection to the mobile application designed in MIT App Inventor, and the integration of the machine learning model trained in TensorFlow for the prediction of gameplay variables. Subsequently, the user was guided through a series of practical tests with the game, during which their reactions and level of understanding were recorded. At the end of the interaction, the standardized SUS (System Usability Scale) questionnaire was administered to measure the user's perception of the system's usability, ease of use, and intuitiveness.

This quantitative assessment was complemented by qualitative comments provided by the user, which facilitated the collection of valuable observations about the interface, game behavior, and intelligent system response.

The process made it possible to verify the viability of the system in a real context and validate the pedagogical and technical objectives set out in the project design.

Fig. 9: Prototype to be evaluated

Fig. 10: Evaluating the project through the survey

Fig. 11: project explanation

A. Description of emotions using a box plot

The box plot presented shows the simulated distribution of emotions of a user when interacting with two versions of an educational game: one with a stuffed animal and another with a robot. The graph is divided by specific emotions (angry, disgusted, fearful, happy, neutral, sad, surprised), and allows for a visual comparison of the average level of each emotion according to the interface used.

It can be observed that the game with the robot generally generates greater emotional intensity in almost all categories. For example, emotions such as angry, happy, neutral, sad, and surprised have higher average values in the version with the robot than with the stuffed animal, indicating greater emotional reactivity. In particular, emotions such as disgusted and fearful show atypically high values in the Robot version, which could indicate specific moments of discomfort or intense surprise that do not occur in the game with the Stuffed Animal.

In contrast, playing with the stuffed animal tends to generate more moderate and homogeneous emotional responses, which can be interpreted as a more stable or controlled emotional experience. This difference suggests that the robot interface, possibly due to its novelty or its capacity for interaction, induces a more intense and varied emotional experience.

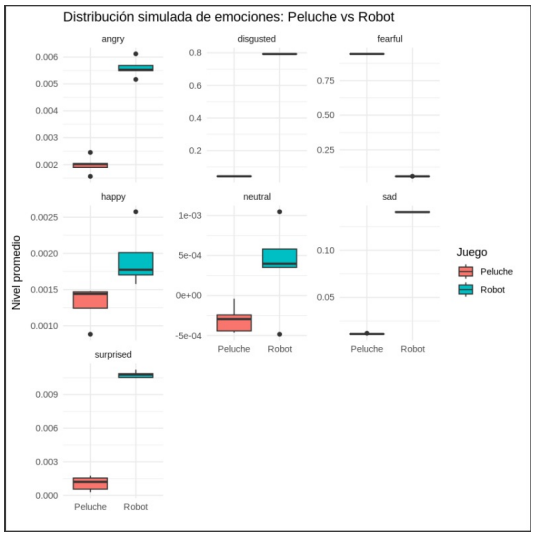

Fig. 12: Box diagram

Overall, the graph shows that the type of interface significantly influences the user's emotional experience, which is crucial for the design of user-centered interactive games, especially if the aim is to stimulate specific emotional responses for educational or therapeutic purposes.

V. Statistical Analysis and Discussion

To evaluate whether there is a significant difference in emotional responses between the two user interfaces—*Capibara Plush* and *CapibaraBot*—a hypothesis test was conducted using the average simulated emotion levels.

Null Hypothesis (H_0): There is no significant difference between the emotional responses generated by the Capibara Plush and the CapibaraBot.

Alternative Hypothesis (H_1): There is a significant difference in emotional responses between the two interfaces.

The result of the test yielded a p-value of **0.04265**. Since this value is below the commonly accepted significance threshold ($\alpha = 0.05$), we reject the null hypothesis. This indicates that the type of interface has a statistically significant effect on the simulated emotional responses.

Figure 12 shows the distribution of average emotion levels across seven categories: *angry*, *disgusted*, *fearful*, *happy*, *neutral*, *sad*, and *surprised*. From the visual analysis, it is evident that:

- The CapibaraBot elicited higher levels of *happy* and *surprised*, suggesting a stronger positive emotional impact.
- Emotions such as *angry* and *sad* also show higher average levels for CapibaraBot, implying increased emotional variability.
- The *neutral* response exhibited greater dispersion for the CapibaraBot group, which may indicate a more ambiguous or mixed emotional perception.

These results suggest that CapibaraBot triggers more intense or variable emotional responses compared to the plush version. This has potential implications for user experience (UX) design in educational and entertainment contexts. However, the increase in emotional intensity is not exclusively positive, and designers should carefully consider how different modalities influence both positive and negative emotional engagement.

Further studies with larger and more diverse samples, as well as qualitative user feedback, are recommended to better understand the underlying causes and to fine-tune the emotional design of interactive agents.

VI. Future Work

- There are several areas that can be improved and expanded for future versions of CapibaraBot. One of the main lines of future work is the inclusion of new game modes that involve more complex cognitive challenges, such as color sequences, direction patterns, or simple logical decisions, which broaden the spectrum of skills stimulated. In addition, there are plans to include progress tracking features, where parents or teachers can access personalized statistics on the child's performance and progress, allowing for closer monitoring of their learning process.
- Another line of development involves incorporating sensors into the robot to enable greater autonomy in interaction, such as avoiding real obstacles or responding to visual and auditory stimuli in the environment. The aim is also to refine the predictive difficulty model, using real data collected from multiple usage sessions, rather than a simulated dataset. This would allow the system to be adjusted to more varied and complex behaviors of real users, increasing its robustness and accuracy.
- A possible integration with augmented reality technologies is planned to enrich the user experience, as well as the internationalization of the system, adapting the content to different languages and cultural contexts. All these projections aim to consolidate CapibaraBot as a low-cost but high-impact educational solution, suitable

for implementation in public and private educational institutions in Latin America.

VII. Conclusions

- The CapibaraBot project represents a significant advance at the intersection of educational robotics, technological accessibility, and artificial intelligence applied to the personalization of children's learning. Through the implementation of a physical toy controlled by a mobile app and powered by a predictive system based on machine learning, a comprehensive tool was developed that responds to the needs of children between the ages of 2 and 5.
- This approach demonstrates that it is possible to design educational technology solutions that are not only effective and functional, but also economical and replicable in resource-limited environments. The use of open technologies such as MIT App Inventor, together with accessible hardware such as the ESP32, allows the project to be scalable and sustainable. The conclusions obtained allow us to affirm that CapibaraBot has the potential to become a transformative tool in early childhood education, contributing to reducing technological gaps and promoting learning through play. Its user-centered design and focus on the child's experience strengthen its relevance and applicability in various Latin American educational contexts.

VIII. Attachments

Dataset construction

```
import pandas as pd
import numpy as np

# Semilla reproducible
np.random.seed(123)
n = 30000

# Variables b sicas aleatorias
bombs_hit = np.random.randint(0, size=n)
projectiles_hit = np.random.randint(0, 1000, size=n)
session_time = np.random.randint(0, 5000, size=n)

# Reglas de dificultad din mica
def calculate_difficulty(bh, ph, st):
    if bh == 2:
        return 'alta'
    elif bh <= 1 and ph >= 10 and st >= 100:
        return 'baja'
    else:
        return 'media'

dificultad = [calculate_difficulty(bh, ph, st) for bh, ph, st in zip(bombs_hit, projectiles_hit, session_time)]

# Mapear salidas din micamente
def map_output(diff):
    if diff == 'baja':
        velocidad = np.random.randint(2, 5)
        frecuencia = np.random.randint(7, 11)
    elif diff == 'media':
        velocidad = np.random.randint(5, 8)
        frecuencia = np.random.randint(4, 7)
    else:  # alta
        velocidad = np.random.randint(8, 11)
        frecuencia = np.random.randint(1, 4)
    return velocidad, frecuencia

outputs = [map_output(d) for d in dificultad]
velocidad_avion, frecuencia_bombas = zip(*outputs)

# Construir DataFrame base
df_large = pd.DataFrame({
    'projectiles_hit': projectiles_hit,
    'session_time': session_time,
    'velocidad_avion': velocidad_avion,
    'frecuencia_bombas': frecuencia_bombas
})

# Crear 100 filas quemadas de caso m nimo
minimo = pd.DataFrame({
    'bombs_hit': [0]*100,
    'projectiles_hit': [0]*100,
    'session_time': [0]*100,
    'velocidad_avion': [2]*100,
    'frecuencia_bombas': [10]*100
})

# Crear 100 filas quemadas de caso m ximo
maximo = pd.DataFrame({
    'bombs_hit': [2]*100,
    'projectiles_hit': [1000]*100,
    'session_time': [5000]*100,
    'velocidad_avion': [10]*100,
    'frecuencia_bombas': [1]*100
})

# Combinar todo
df_final = pd.concat([df_large, minimo, maximo], ignore_index=True)

# Guardar CSV
csv_large_path = "juego_dataset_30200.csv"
df_final.to_csv(csv_large_path, index=False)

\end{verbatim}
Performance evaluation
\begin{verbatim}
import pandas as pd
import numpy as np
from sklearn.model_selection import train_test_split
import matplotlib.pyplot as plt
import ipywidgets as widgets

# 1. Cargar dataset generado previamente
df = pd.read_csv("juego_dataset_30200.csv")

# 2. Separar variables de entrada y salida
X = df[['bombs_hit', 'projectiles_hit', 'session_time']]
y = df[['velocidad_avion', 'frecuencia_bombas']]

# 3. Dividir en entrenamiento y prueba
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# 4. Entrenar modelo
model = RandomForestRegressor(n_estimators=100, random_state=42)
model.fit(X_train, y_train)

# 5. Evaluar rendimiento
y_pred = model.predict(X_test)

mae_total = mean_absolute_error(y_test, y_pred)
mae_velocidad = mean_absolute_error(y_test['velocidad_avion'], y_pred[:, 0])
mae_frecuencia = mean_absolute_error(y_test['frecuencia_bombas'], y_pred[:, 1])

print("\n Evaluaci n del modelo:")
print(f"MAE total promedio: {mae_total:.2f}")
print(f"MAE velocidad a v i n: {mae_velocidad:.2f}")
print(f"MAE frecuencia bombas: {mae_frecuencia:.2f}")

```

```
# 6. Visualizar predicciones reales vs predichas
plt.figure(figsize=(12, 5))

# Subplot 1: Velocidad del avi n
plt.subplot(1, 2, 1)
plt.scatter(y_test['velocidad_avion'], y_pred[:, 0], alpha=0.5, color='blue')
plt.plot([2, 10], [2, 10], 'k--')
plt.xlabel("Velocidad real")
plt.ylabel("Velocidad predicha")
plt.title("Velocidad del avi n")
plt.grid(True)

# Subplot 2: Frecuencia de bombas
plt.subplot(1, 2, 2)
plt.scatter(y_test['frecuencia_bombas'], y_pred[:, 1], alpha=0.5, color='orange')
plt.plot([1, 10], [1, 10], 'k--')
plt.xlabel("Frecuencia real")
plt.title("Frecuencia de bombas")
plt.grid(True)

plt.tight_layout()
plt.show()

# 7. Widgets para ingresar valores
bombs_hit_input = widgets.BoundedIntText(
    value=1, min=0,step=1, description=' Bombas:'
)

projectiles_hit_input = widgets.BoundedIntText(
    value=5, min=0, max=1000, description='Proyectiles:'
)

session_time_input = widgets.BoundedIntText(
    value=100, min=0, max=5000, step=1, description=' Tiempo (s):'
)

output = widgets.Output()

def hacer_prediccion(change=None):
    bh = bombs_hit_input.value
    st = session_time_input.value

    nuevos_datos = pd.DataFrame([{
        'bombs_hit': bh,
        'projectiles_hit': ph,
        'session_time': st
    }])

    prediccion = model.predict(nuevos_datos)[0]

    with output:
        output.clear_output()
        print(" Predicci n basada en los datos ingresados:")
        print(f"Velocidad del avi n: {round(prediccion[0])} (2=f cil, 10=dif cil)")
        print(f"Frecuencia de bombas: {round(prediccion[1])} (10=f cil, 1=dif cil)")


# Bot n
btn = widgets.Button(description=" Predecir dificultad")
btn.on_click(hacer_prediccion)

# Mostrar interfaz
display(bombs_hit_input, projectiles_hit_input, session_time_input, btn, output)
\end{verbatim}
System training
\begin{verbatim}
import pandas as pd
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.metrics import mean_absolute_error
import joblib

# 1. Cargar dataset
df = pd.read_csv("juego_dataset_30200.csv")

# 2. Variables de entrada (X) y salida (y)
X = df[['bombs_hit', 'projectiles_hit', 'session_time']]
y = df[['velocidad_avion', 'frecuencia_bombas']]

# 3. Divisi n en entrenamiento y prueba
X_train, X_test, y_train = train_test_split(X, y, test_size=0.2, random_state=42)

# 4. Entrenamiento del modelo
modelo = RandomForestRegressor(n_estimators=100, random_state=42)
modelo.fit(X_train, y_train)

# 5. Evaluaci n del modelo
y_pred = modelo.predict(X_test)
mae_velocidad = mean_absolute_error(y_test['velocidad_avion'], y_pred[:, 0])
mae_frecuencia = mean_absolute_error(y_test['frecuencia_bombas'], y_pred[:, 1])

print(" Evaluaci n del modelo:")
print(f" MAE total promedio: {mae_total:.2f}")
print(f"MAE velocidad avi n: {mae_velocidad:.2f}")
print(f"MAE frecuencia bombas: {mae_frecuencia:.2f}")

# 6. Exportar modelo como archivo .joblib
joblib.dump(modelo, "modelCapibara.joblib")
```

Listing 1: Ejemplo de código en Python

REFERENCES

[1] Alexis Boulch, Yann Antoine, Guy Rousseau, and otros. Porting ardupilot to esp32: Towards a universal open-source architecture for agile and easily replicable multi-domains mapping robots. In *ISPRS - International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, volume 43, pages 933–940, 2020.

[2] Wei Chen, Andrew Huey Ping Tan, Yang Luo, and otros. Modular robotic controllers with ad-hoc esp32 wireless mesh network and embedded aes security. In *Selected Proceedings from the 2nd International Conference on Intelligent Manufacturing and Robotics, ICIMR 2024 - Advances in Intelligent Manufacturing and Robotics*, pages 376–390. Springer, 2024.

[3] Israel Marcelo Moya Herrera and Arlena Omayra Zambrano Cuadro. *Diseño e implementación de un robot cuadrúpedo para competencias con control autónomo o manual mediante una aplicación móvil*. PhD thesis, Universidad Politécnica Salesiana, 2025. Disponible en: http://dspace.ups.edu.ec/handle/123456789/30221.

[4] M D Patil, Om Sandeep Ghadge, Siddharth Satish Mohite, and Omkar Santosh Shinde. 5-dof robotic arm using esp32. *Journal of Emerging Technologies and Innovative Research*, 11(7), 2024.

[5] Yan Zhang and Lei Zhang. Research on education robot control system based on esp32. *Journal of Education and Educational Research*, 7(2):299–306, 2024.