REVISTA PERSPECTIVAS

VOL. 7, N°2 / JULIO - DICIEMBRE 2025 / e-ISSN: 266-6688

DOI: https://doi.org/10.47187/perspectivas.7.2.237

RECONOCIMIENTO DE LENGUAJE DE SEÑAS PARA EL CONTROL DE DISPOSITIVOS ELÉCTRICOS

Sign Language Recognition for Electrical Device Control

Jefferson Ribadeneira Ramírez [1]	jefferson.ribadeneira@espoch.edu.ec
Robert Rodríguez Loaiza [2]	robert.rodriguez@espoch.edu.ec
Kevin Saavedra Delgado [3]	svdknd96m29z605m@studenti.unical.it
Ana Logroño Noboa [4]	lucia.logrono@unach.edu.ec

[1 2] Facultad de Informática y Electrónica, Escuela Superior Politécnica de Chimborazo, Riobamba, Ecuador´

[3] Universidad de Calabria, Arcavacata di Rende, Italia

[4] Dirección de Postgrado, Universidad Nacional de Chimborazo, Riobamba, Ecuador

ABSTRACT

This research paper presents the implementation of a dactylographic sign language recognition system using artificial intelligence applied to control electrical devices. The system aims to enhance interaction between humans and electrical devices, particularly for individuals with hearing impairments, by providing a natural and intuitive method for interaction. The system utilizes computer vision techniques to recognize hand gestures in dactylographic sign language, and a microcontroller-based circuit controls the electrical devices. The system performance was evaluated in terms of accuracy, response time, and usability, yielding promising results for future applications in industry and medicine.

Keywords: Dactylographic Sign Language Recognition, Artificial Intelligence, Artificial Vision, Microcontroller-Based Circuit, Electrical Devices, Wireless Communication, Remote Control.

RESUMEN

Este trabajo de investigación presenta la implementación de un sistema de reconocimiento de lenguaje de señas dactilográfico mediante inteligencia artificial aplicada al control de dispositivos eléctricos. El sistema tiene como objetivo mejorar la interacción entre humanos y dispositivos eléctricos, particularmente para personas con discapacidad auditiva, proporcionando un método de interacción natural e intuitivo. El sistema utiliza técnicas de visión por computadora para reconocer gestos con las manos en lenguaje de señas dactilográfico y un circuito basado en microcontrolador controla los dispositivos eléctricos. El rendimiento del sistema se evaluó en términos de precisión, tiempo de respuesta y usabilidad, lo que arrojó resultados prometedores para futuras aplicaciones en la industria y la medicina.

Palabras Clave: Reconocimiento de Lenguaje de Señas Dactilográfico, Inteligencia Artificial, Visión Artificial, Circuito Basado en Microcontrolador, Dispositivos Eléctricos, Comunicación Inalámbrica, Control Remoto.

I. Introducción

In recent years, the development of assistive technologies has achieved great importance, especially for people with disabilities. One of the areas that has seen significant progress is the sign language recognition systems to improve communication and control of electrical devices. In this context, the implementation of a system to control electrical devices using dactylographic sign language recognition through artificial vision represents a significant step forward in this field [1].

Fecha de Recepción: 22/04/2025. **Fecha de Aceptación:** 20/06/2025. **Fecha de Publicación:** 07/07/2025

This type of system employs advanced machine learning techniques to accurately recognize and interpret sign language gestures.

The disability issues are addressed by all governments, which implement different strategies to carry out this problem, particularly focusing on access to various areas such as education, health, transportation, etc. [2]. Therefore, the use of technological tools provides several alternatives for its treatment. Nowadays, novel techniques, as Artificial Intelligence (AI), are applied to computer vision and language recognition, and these has the potential of contribute significantly to these efforts, enabling the implementation of many systems which could be use by people whose abilities are not within the "normal" range [3].

In this perspective, hand gesture recognition is a research area that has captured the attention of many people for the development of Human-Computer Interaction (HCI) applications, including virtual reality, augmented reality, games, educational applications, among others. [5]. In other hand, remote devices control and automation is a current trend, driven by the implementation of Internet of Things (IoT) technologies and its potential to connect all types of devices. High-impact technology such as Computer Vision is another current trend, which creates a broad field of innovative applications, in which real-time image and video processing allow for control and visualization of large amount data on internet. The main applications that are being developed with these two trends are focused on education, medicine, smart buildings, people and vehicle surveillance systems, etc. These types of applications could improve the quality of life for its users. However, their development requires infrastructure that allows the convergence of different technologies and devices, especially those that handle the phases of image acquisition and processing [6] [7].

This study presents the design and implementation of a system which uses trending techniques to recognize and interpret sign language gestures accurately. This type of system improves the interaction between people with hearing disabilities and electrical devices. The paper presents methodology used for the development of system and results obtained from the tests carried out.

II. Methodology

A. Equipment and Materials

1) Hardware: The hardware used in this study includes the following components:

- ASUS STRIX RYZEN 7 laptop with 16 GB of RAM and a 512 GB SSD.
- GPU RTX 3050.
- ESP8266 Module Relay.
- 110V Sound Alarm.
- Light bulb.
- Electric Lock.
- 5V Regulator.
- Webcam.

2) Software: The software and operating systems utilized in this research are as follows:

- Windows 11 as the operating system.
- Arduino IDE version 1.18.19 for hardware programming.
- Anaconda for Python environment management.
- PyCharm Community for Python development.
- Mosquitto for MQTT (Message Queuing Telemetry
- Transport) messaging protocol.
- Jupyter Notebook for interactive data analysis and visualization.

B. System Design

Figure 1 shows the architecture of the system, which is detailed in the following steps:

1) Data Acquisition: The system comprises a computer connected to a camera for signal capture.

2) Signal Processing: Signal processing is performed using the MediaPipe Hand Tracking algorithm, customized for this project within PyCharm Community. MediaPipe utilizes pre-trained artificial neural networks for real-time image processing. Training is conducted in Jupyter

Notebook using previously collected images.

3) Data Transmission: After signal detection, the system transmits data to an MQTT broker server, serving as an intermediary between subscribers and publishers. The system employs three topics, each associated with an end device (light bulb, siren or lock). When a recognized gesture is detected, a signal is sent to respective topics to activate the corresponding electrical device.

4) Device Control: An ESP8266 module, programmed via Arduino software, receives the signal from the MQTT server through a Wi-Fi connection. The ESP8266 module then amplifies the signal using a relay module containing a 2N2222 transistor. This relay module subsequently activates the relay, which controls the electrical device. Both the relay module and the ESP8266 module are powered by a 5V regulator.

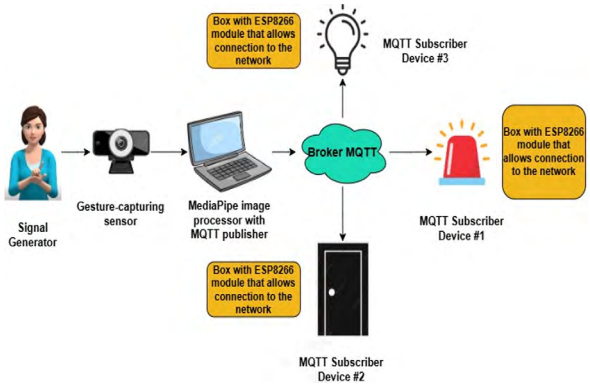


Fig. 1: *System Architecture*

C. Tailored Model for Image Capture

A personalized model was developed to capture images for both training and real-time detection purposes. Within the same script. Figure 2 show six different labels were selected for this project to enable or disable devices through gesture recognition. Following the selection of signs for activating or deactivating devices, a CSV document is created, wherein the labels for each sign are recorded to be subsequently exhibited on the screen, as show Figure 3(b). These signs were extracted from the fundamental glossary of Ecuadorian Sign Language, which was established by Vice Presidency of the Republic, Ecuador.

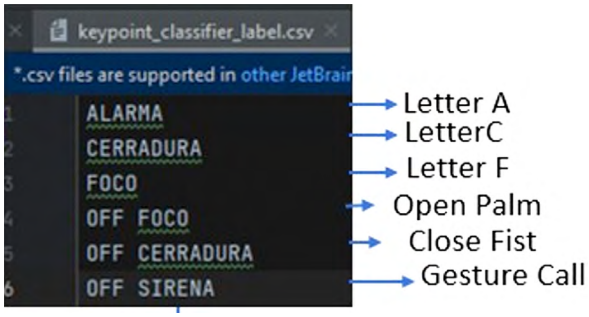


Fig. 2: *Labels for training*

In the primary script, a function was implemented to toggle between keypoint capture mode and exit mode using a variable called" mode" as show the Figure 3(a). During program execution, the" K" key, stored in the" key" variable, initiates keypoint capture mode. Furthermore, the" number" variable, represents the maximum number of images, facilitates the acquisition of sufficient images for training within the program.

The variable "number" is bounded to values ranging from 0 to 9, with 10 being the maximum limit for labels stored in the program. This limit can be extended to incorporate more images into the program as required. The images are captured using the same keys that denote the maximum number of labels. For example, the "0" key is used to capture images of the first sign, which represents the "A" sign that corresponds to sound alarm, while the "5" key is utilized to capture the sign that deactivates the alarm.

```
def select_mode(key, mode):
    number = -1
    if 48 <= key <= 57:  # 0 ~ 9
        number = key - 48
    #if key == 110:  # n
        #mode = 0
    if key == 107:  # k
        mode = 1
    return number, mode
```

(a) *Function to capture keypoints of the hand*

```
0,0.0,0.0,-0.3695652173913043,-0.11304347826086956,-0.6347826086956522,
0,0.0,0.0,-0.3565217391304348,-0.08695652173913043,-0.6260869565217392,
0,0.0,0.0,-0.35807860262008734,-0.08733624454148471,-0.6157205240174672,
0,0.0,0.0,-0.37168141592920356,-0.0752212389380531,-0.6283185840707964,
0,0.0,0.0,-0.3744292237442922,-0.045662100456621,-0.6118721461187214,-0.
0,0.0,0.0,-0.37037037037037035,-0.04166666666666664,-0.5879629629629629
0,0.0,0.0,-0.3548387096774194,-0.009216589861751152,-0.5622119815668203,
0,0.0,0.0,-0.33613445378151263,-0.10084033613445378,-0.6218487394957983,
0,0.0,0.0,-0.3217391304347826,-0.10434782608695652,-0.6086956521739131,
```

(b) *Function to capture keypoints of the hand*

Fig. 3: *Tailored Model.*

D. Training Model

The model training process is achieved by implementing a sequential neural network, which utilizes a previously stored dataset containing all the key points of image captures used for training. The TensorFlow neural network model comprises five layers, using a sequence of consecutive layers to process the input data.

The first layer, **tf.keras.layers.Input**', accepts an input of shape (21*2), which refers to the size of a tensor or vector with 42 elements. The multiplication of 21 by 2 represents the X and Y coordinates of each of the 21 hand reference points.

Following this, a Dropout layer with a rate of 0.2 and 42 neurons is included to prevent overfitting. A dense layer with 20 neurons and a ReLU activation function is then introduced. This is followed by another Dropout layer with a rate of 0.4 and 20 neurons. A further dense layer with 10 neurons and a ReLU activation function is included. Finally, the last dense layer comprises six neurons and a softmax activation function for classification purposes, an Artificial Neural Network (ANN) is illustrated in Figure 4.

The training process was concluded by adjusting various parameters for execution and utilizing the '**fit**' method of the model to train the neural network. Specifically, the training was performed using the '**X_train**' and '**Y_train**' data for a total of 1000 epochs, with a batch size of 128 samples. Validation was carried out using the '**X_test**' and '**Y_test**' data.

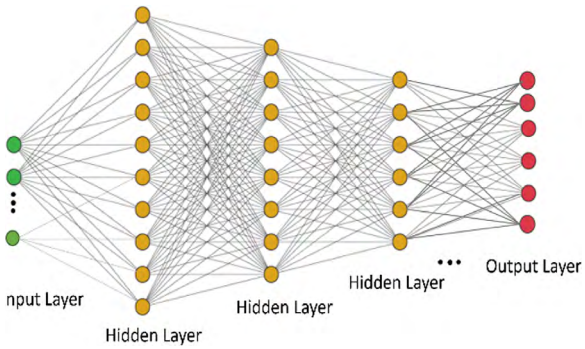


Fig. 4: *Artificial Neural Network architecture*

Additionally, two callbacks, namely '**cp_callback**' and '**es_callback**', were implemented to enable supplementary tasks during the training phase, such as weight preservation or early termination. It is noteworthy that the Adam optimizer was employed to update the weights during training process, while the loss function was utilized to measure the discrepancy between model predictions and the true labels. The evaluation of the performance of the model was based on the accuracy metric, which assesses the proportion of correct predictions relative to the true labels.

E. Real-Time detection

The detection process involves a loop that continues until the ESC key is pressed. It calculates the bounding box around detected hands and retrieves the associated landmarks. These landmark coordinates are normalized for ease of use in training a machine learning model. Additionally, the function "loggin_csv" is called to save the preprocessed landmark data in a CSV file, which will be used in the creation of a machine learning model. This process ensures comprehensive data processing and storage for future use, as shown in Figure 5.

```
if results.multi_hand_landmarks is not None:
    for hand_landmarks, handedness in zip(results.multi_hand_landmarks,
                                          results.multi_handedness):
        # Calculando cuadro de delimitación
        brect = calc_bounding_rect(debug_image, hand_landmarks)
        # Landmark calculation
        landmark_list = calc_landmark_list(debug_image, hand_landmarks)
        # Conversion to relative coordinates / normalized coordinates
        pre_processed_landmark_list = pre_process_landmark(
            landmark_list)
        pre_processed_point_history_list = pre_process_point_history(
            debug_image, point_history)
        # Guardar en el archivo que contiene los datos
        logging_csv(number, mode, pre_processed_landmark_list,
                    pre_processed_point_history_list)
```

Fig. 5: *Real-time detection*

F. Data Transmission

To facilitate data transmission through MQTT, a MQTT client instance is first established. Subsequently, the custom function 'keypoint_ classifier' is utilized to discern the hand signal by leveraging the hand landmarks. The outcome of this classification is stored in the variable 'hand_sign_id'. Once this outcome is acquired, the variable is subjected to conditional statements

to effectuate the activation or deactivation of the respective electrical devices. Notably, each device is activated in a distinct manner and the cessation of all devices is achieved by employing the character 'F', this phase is illustrated in Figure 6.

G. Data Reception

The final device incorporates an ESP8266 module, which connects to a Wi-Fi network to receive data from the MQTT server, in this case, "test.mosquitto.org". The final device is connected to Pin D7, which is defined as pin 13 in the programming. Pin D7 is linked to a relay module, which utilizes a 2N2222 transistor as an amplifier to activate the relay and control the electrical device. The ESP8266 module operates at a baud rate of 115200 baud/s and communicates via port 1883, utilizing key functions for data reception, as illustrated in Figure 7.

```
# Clasificación de los signos de mano
client = mqtt.Client()
hand_sign_id = keypoint_classifier(pre_processed_landmark_list)
print(hand_sign_id)

if hand_sign_id == 0:
    client.connect("test.mosquitto.org", 1883, 60)
    client.publish("prueba/tres", "A")
if hand_sign_id == 1:
    client.connect("test.mosquitto.org", 1883, 60)
    client.publish("prueba/dos","C")
```

Fig. 6: *MQTT connection and data transmission*

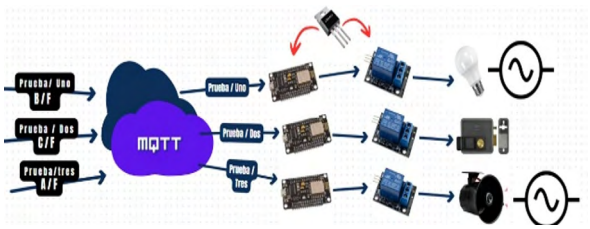


Fig. 7: *Data Reception*

III. Results

During the training phase, a dataset of approximately 8572 images was acquired. Each image was captured while the corresponding digits were entered by keyboard input. Through the utilization of MediaPipe landmarks and the CSV library, the coordinates of landmarks for each captured image were promptly extracted and stored in a tabular format. This approach of collecting point-based representations obviated the need to store a large volume of actual images, enhancing efficiency and reducing the storage requirements of the project.

The collected data, organized as landmark points, was categorized into six distinct classes. These classes were assigned to specific hand gestures for the activation and deactivation of electrical devices. Class 0 denoted the activation of a siren, comprising a total of 1371 captured images. Class 1 represented the activation of a lock, with a dataset of 1907 images. The activation of a light bulb fell under class 2, with 1904 images captured. Conversely, class 3 involved the deactivation of light bulb, consisting of 1438 images. Class 4 corresponded to the deactivation of lock, with 1067 images collected. Lastly, class 5 denoted the deactivation of siren, encompassing 885 images. The determination of image quantities per class was accomplished through an iterative trial-and-error process, guided by the analysis of confusion matrix to optimize the dataset distribution and improve the overall system performance.

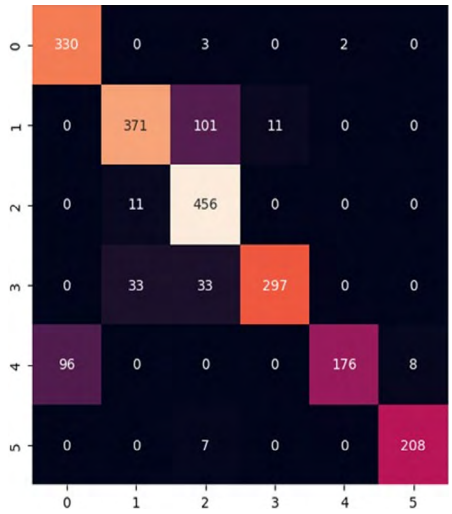
A. Training

The training process takes approximately 27 seconds, while the execution of script containing the training and other evaluation parameters together lasts around 36 seconds. During the training, 17 batches of data of size 128 were evaluated, each evaluation took 2 ms/step. The average loss of the model on test dataset during evaluation is 0.4134, indicating the disparity between predictions of the model and actual values. The average accuracy of the model on test dataset during evaluation is 0.8577, signifying the precision of model predictions in relation to the actual values. The model demonstrates an accuracy of 85.77%.

B. Matrix Confusion

Figure 8(a) shows the confusion matrix reveals instances of misclassification between Class 0, associated with activating the alarm, and Class 4, corresponding to the deactivation of lock. Similarly, Class 1, representing the activation of lock, exhibits confusion with both the activation of the light (Class 2) and the deactivation of the

light bulb (Class 3). Notably, these confusions align with the similarity of gestures between these classes. Moreover, the classification performance report summarizes the metrics for performance evaluation on the test dataset.



(a) *Matrix of confusion*

Classification Report

	precision	recall	f1-score	support
0	0.77	0.99	0.87	335
1	0.89	0.77	0.83	483
2	0.76	0.98	0.85	467
3	0.96	0.82	0.89	363
4	0.99	0.63	0.77	280
5	0.96	0.97	0.97	215
accuracy			0.86	2143
macro avg	0.89	0.86	0.86	2143
weighted avg	0.88	0.86	0.86	2143

(B) *Classification Report*

Fig. 8: *Classification Results*

The precision refers to the fraction of instances classified correctly for a class, while recall pertains to the fraction of class instances that were correctly classified. The F1 score represents a combined measure of precision and recall. Both the macro average and weighted average provide the meaning of these metrics across all classes. The support indicates the number of samples in the test dataset that belong to each class, the classification report is illustrated in Figure 8(b).

C. Real-time detection

Real-time detection provides high precision and detection speed in milliseconds, with the program running at 14 to 20 frames per second (FPS) when not connected to MQTT. It's worth noting that the detection is being performed on the CPU since the MediaPipe real-time detection algorithm does not support GPU execution. Additionally, the training process was also conducted using the CPU.

The relay is activated when executing the gesture corresponding to the activation of the electrical device and is deactivated when executing the gesture corresponding to the deactivation of the electrical device.

1) Alarm activation: For Class 0 detection, which cor responds to the activation of the sound alarm, correct sign placement yields 100% precision. However, if the hand is rotated into a fist-like shape, it may cause confusion. In 10 detection's, all predictions were accurate. This activation is illustrated in the Figure 9.

2) Electric Lock Activation: Class 1 detection, corresponding to the activation of the lock as show the Figure 10, exhibits an 89% precision according to the confusion matrix. However, the detection is almost perfect except when the wrist is lifted upwards or slightly tilted towards the right.

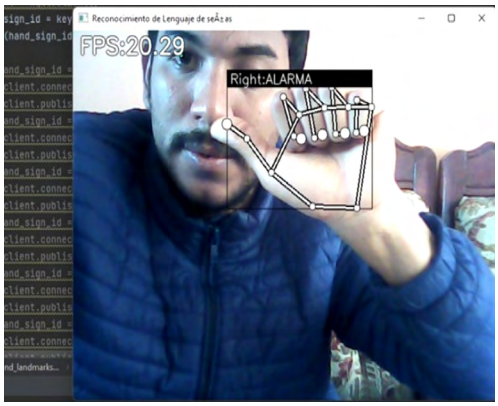


Fig. 9: *Alarm Activation*

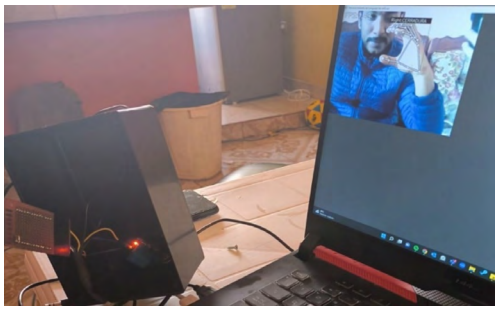


Fig. 10: *Electric lock activation*

3) Light bulb Activation: For Class 2 detection, corresponding to the activation of the light bulb, as illustrated in Figure 11. A 100% detection is achieved unless one of the three fingers of the sign is lowered, in which case it is confused with the lock.

4) Light bulb Desactivation: Class 3, corresponding to the deactivation of the light bulb, as show in Figure 12, provides a 96% precision according to the confusion matrix. However, the detection is almost perfect, except for rare cases when the hand is tilted to the left or right. Out of 12 tilted detection's, only 1 was incorrect.

5) Electric Lock Desactivation: The detection of Class 4, corresponding to the deactivation of the lock, illustrated in Figure 13, yields remarkable results with a precision of 99%. Out of a total of 15 detection's performed, each one successfully identified this particular signal.

Fig. 11: *Light bulb activation*

Fig. 12: *Light bulb desactivation*

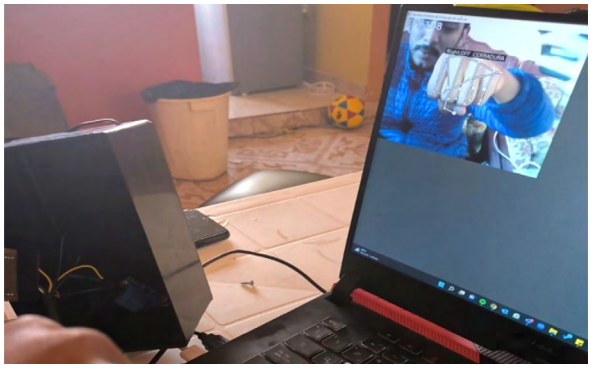


Fig. 13: *Electric lock desactivation*

6) Alarm desactivation: Class 5, corresponding to the de activation of the siren, exhibits an impressive 96% precision, as shown in the confusion matrix. Out of the 10 detection's performed, all 10 were accurately recognized for this particular gesture, this desactivation is illustrated in Figure 14.

Fig. 14: *Alarm deactivation*

IV. Conclusions

The training process of the gesture recognition model with MediaPipe is more straightforward and efficient, thanks to the utilization of specific functions and statements. This reduction in training time, down to 30 seconds for 8572 images, demonstrates the advantages of Keypoint Classifier Model over traditional models. It allows data to be trained as points in a spreadsheet, accelerating the process. Additionally, the implementation of the EarlyStopping object prevents resource waste and overfitting.

The detection speed of the system ranges from 14 to 20 FPS without an MQTT server connection

and from 2 to 5 FPS with an active connection due to the use of TCP protocol. However, it has been demonstrated that this level of speed is sufficient for real-time control of electrical devices.

In summary, the proposed system represents a significant contribution to computer vision and accessibility for indi viduals with disabilities. It enhances their quality of life and autonomy by enabling gesture-based control of electrical devices.

V. Referencias

[1] ”Discapacidad: lo que todos debemos saber”, Revista do Instituto de Medicina Tropical de S˜ao Paulo, vol. 48, n.º 3, pp. 146, junio de 2006. Accedido el 31 de julio de 2023. [En l´ınea]. Disponible: https://doi.org/10.1590/s0036-46652006000300015

[2] S. K y P. R, ”Sign Language Recognition System Using Neural Net works”, Int. J. Res. Appl. Sci. Eng. Technol., vol. 10, n.º 6, p. 827–831, junio de 2022. Accedido el 1 de agosto de 2023. [En l´ ınea]. Disponible: https://doi.org/10.22214/ijraset.2022.43787

[3] M. Ishihara y M. Tsuda, ”Hearing support system for the hear ing impaired”, en 2021 IEEE 3rd Global Conf. Life Sci. Tech nol. (LifeTech), Nara, Japan, 9–11 de marzo de 2021. IEEE, 2021. Accedido el 1 de agosto de 2023. [En línea]. Disponible: https://doi.org/10.1109/lifetech52111.2021.9391875

[4] D. K. Choudhary, R. Singh y D. Kamthania, ”Sign Language Recog nition System”, SSRN Electron. J., 2021. [En l´ ınea]. Disponible: https://doi.org/10.2139/ssrn.3832151.

[5] N. Norouzi, G. Bruder, B. Belna, S. Mutter, D. Turgut y G. Welch, A Systematic Review of the Convergence of Augmented Reality, Intelligent Virtual Agents, and the Internet of Things, en Artificial Intelligence in IoT. Cham: Springer International Publishing, 2019, p. 1–24. Disponible: https://doi.org/10.1007/978-3-030-04110-6 1.

[6] X. Liang,” Internet of Things and its applications in libraries: a literature review”, Library Hi Tech, vol. 38, n.º 1, p. 67–77, agosto de 2018. Disponible: https://doi.org/10.1108/lht-01-2018-0014.

[7] A. Muñoz, J. C. Augusto, V. Tam y H. Aghajan,” Artificial intelligence for IoT systems”, J. Ambient Intell. Smart Environ., vol. 12, n.º 1, pp. 1, enero de 2020. Accedido el 1 de agosto de 2023. [En l´ ınea]. Disponible: https://doi.org/10.3233/ais-200548