

**Prototype Arduino Python**

**Members:**

Griselda Cortes Barrera

Claudia Teresa González Alfaro

Jhacer Kharen Ruiz Garduño

Jorge Abraham Jiménez Alfaro

**Content**

[Python Instructions Manual 3](#_Toc174890623)

[Some execution test results. 9](#_Toc174890625)

[Arduino Instructions Manual (LEDs) 10](#_Toc174890627)

[Diagram. 10](#_Toc174890628)

[Arduino Code 10](#_Toc174890630)

[Python Code 12](#_Toc174890631)

[Virtual Headset 13](#_Toc174890632)

[Arduino Instructions Manual (Cart) 16](#_Toc174890633)

[Arduino Code 17](#_Toc174890635)

**Figure index**

[Figure 1 Pip install websocket-client 3](file:///D:\Manuales%20de%20Artículo\Arduino%20Python%20Prototype%20%20%20V2.inglés.docx#_Toc174898002)

[Figure 2 Pip install Python-dipatch 3](file:///D:\Manuales%20de%20Artículo\Arduino%20Python%20Prototype%20%20%20V2.inglés.docx#_Toc174898003)

[Figure 3 Pip install cortex2 3](file:///D:\Manuales%20de%20Artículo\Arduino%20Python%20Prototype%20%20%20V2.inglés.docx#_Toc174898004)

[Figure 4 Pyhton -m pip install pyserial 4](file:///D:\Manuales%20de%20Artículo\Arduino%20Python%20Prototype%20%20%20V2.inglés.docx#_Toc174898005)

[Figure 5 Pyhton -m pip install serial 4](file:///D:\Manuales%20de%20Artículo\Arduino%20Python%20Prototype%20%20%20V2.inglés.docx#_Toc174898006)

[Figure 6 Python File 5](#_Toc174898007)

[Figure 7 Extension Installation 5](#_Toc174898008)

[Figure 8 live\_advance Code 6](#_Toc174898009)

[Figure 9 Serial Class 6](#_Toc174898010)

[Figure 10 Write Method 7](#_Toc174898011)

[Figure 11 Access Keys 7](#_Toc174898012)

[Figure 12 Trained Profile Name 7](#_Toc174898013)

[Figure 13 Profile Movements 8](#_Toc174898014)

[Figure 14 Client Name 8](#_Toc174898015)

[Figure 15 Emotiv Launcher Error 8](file:///D:\Manuales%20de%20Artículo\Arduino%20Python%20Prototype%20%20%20V2.inglés.docx#_Toc174898016)

[Figure 16 EMOTIV Headset Response 9](#_Toc174898017)

[Figure 17 Execution Tests 9](#_Toc174898018)

[Figure 18 Arduino Connection Diagram 10](#_Toc174898019)

[Figure 19 Arduino Code 11](file:///D:\Manuales%20de%20Artículo\Arduino%20Python%20Prototype%20%20%20V2.inglés.docx#_Toc174898020)

[Figure 20 Variable Definitions 12](#_Toc174898021)

[Figure 21 Serial Port Definition 13](file:///D:\Manuales%20de%20Artículo\Arduino%20Python%20Prototype%20%20%20V2.inglés.docx#_Toc174898022)

[Figure 22 Add Headset 13](#_Toc174898023)

[Figure 23 Headset Types 14](#_Toc174898024)

[Figure 24 Headset On 14](#_Toc174898025)

[Figure 25 Headset Connection 14](#_Toc174898026)

[Figure 26 Connection Validation 15](#_Toc174898027)

[Figure 27 Headset Configuration 15](#_Toc174898028)

[Figure 28 Movement Configuration 15](#_Toc174898029)

[Figure 29 Movement Types 16](#_Toc174898030)

[Figure 30 Design Diagram 16](#_Toc174898031)

[Figure 31 Arduino Configuration Code (Part 1) 17](file:///D:\Manuales%20de%20Artículo\Arduino%20Python%20Prototype%20%20%20V2.inglés.docx#_Toc174898032)

[Figure 32 Arduino Configuration Code (Part 2) 18](#_Toc174898033)

[Figure 33 Variable Definition 19](#_Toc174898034)

[Figure 34 Screen Error 20](file:///D:\Manuales%20de%20Artículo\Arduino%20Python%20Prototype%20%20%20V2.inglés.docx#_Toc174898035)

# Python Instructions Manual

First, download the libraries needed for Python, which is the environment where the program connecting to our Arduino prototype will run.

Ensure you have the latest version of Python; if not, you can download it from the following link: [Python Download](https://www.python.org/downloads/)

También se debe contar con Visual Studio Code o Visual Studio.

To begin, open CMD (preferably run CMD as administrator) and execute the following commands:

1. **pip install websocket-client**

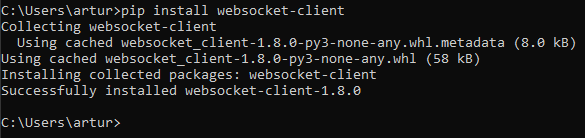


Figure 1 Pip install websocket-client

1. **pip install Python-dipatch**

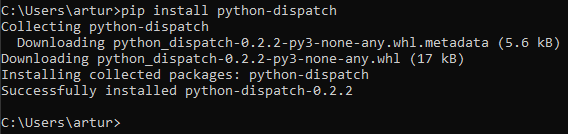


Figure 2 Pip install Python-dipatch

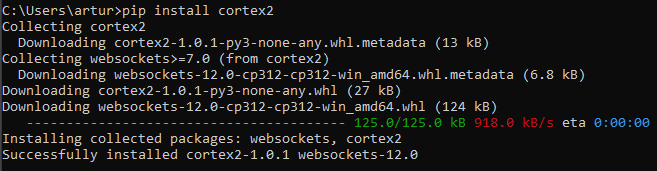
1. **pip install cortex2** (Also install pip install cortex in case the program does not run.)

Figure 3 Pip install cortex2

1. **pyhton -m pip install pyserial**

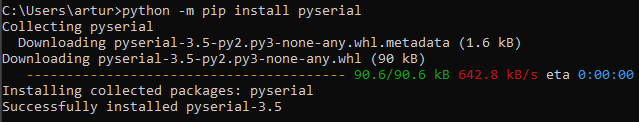


Figure 4 Pyhton -m pip install pyserial

1. **pyhton -m pip install serial**

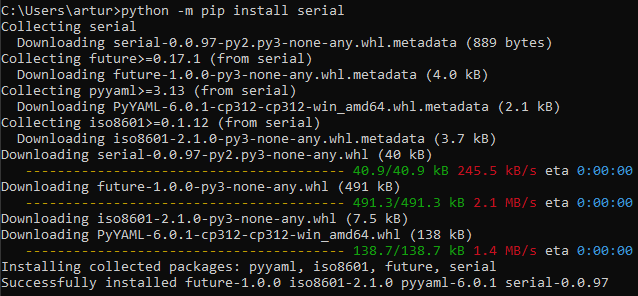


Figure 5 Pyhton -m pip install serial

All installations should indicate “Successfully installed,” which means the libraries were installed correctly and you can proceed with the procedure.

Once you have the necessary Python libraries, modify the code found in the Python folder inside the main folder named cortex-example-master.

Use Visual Studio Code, drag and drop the Python folder into the editor to open it directly, or open the folder from File.

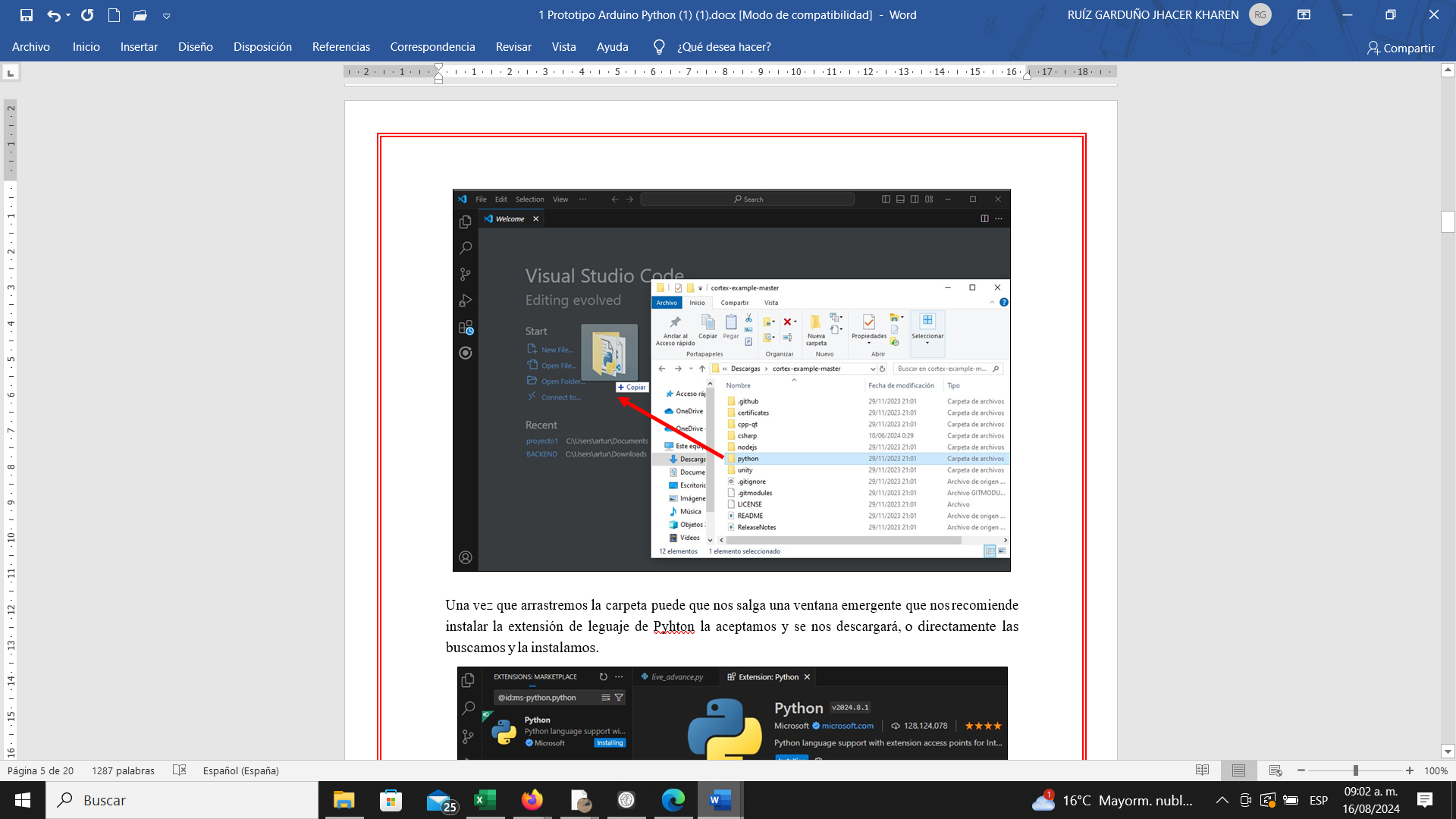


Figure 6 Python File

A pop-up window may appear recommending installing the Python language extension. Accept it to download or search for and install it directly.

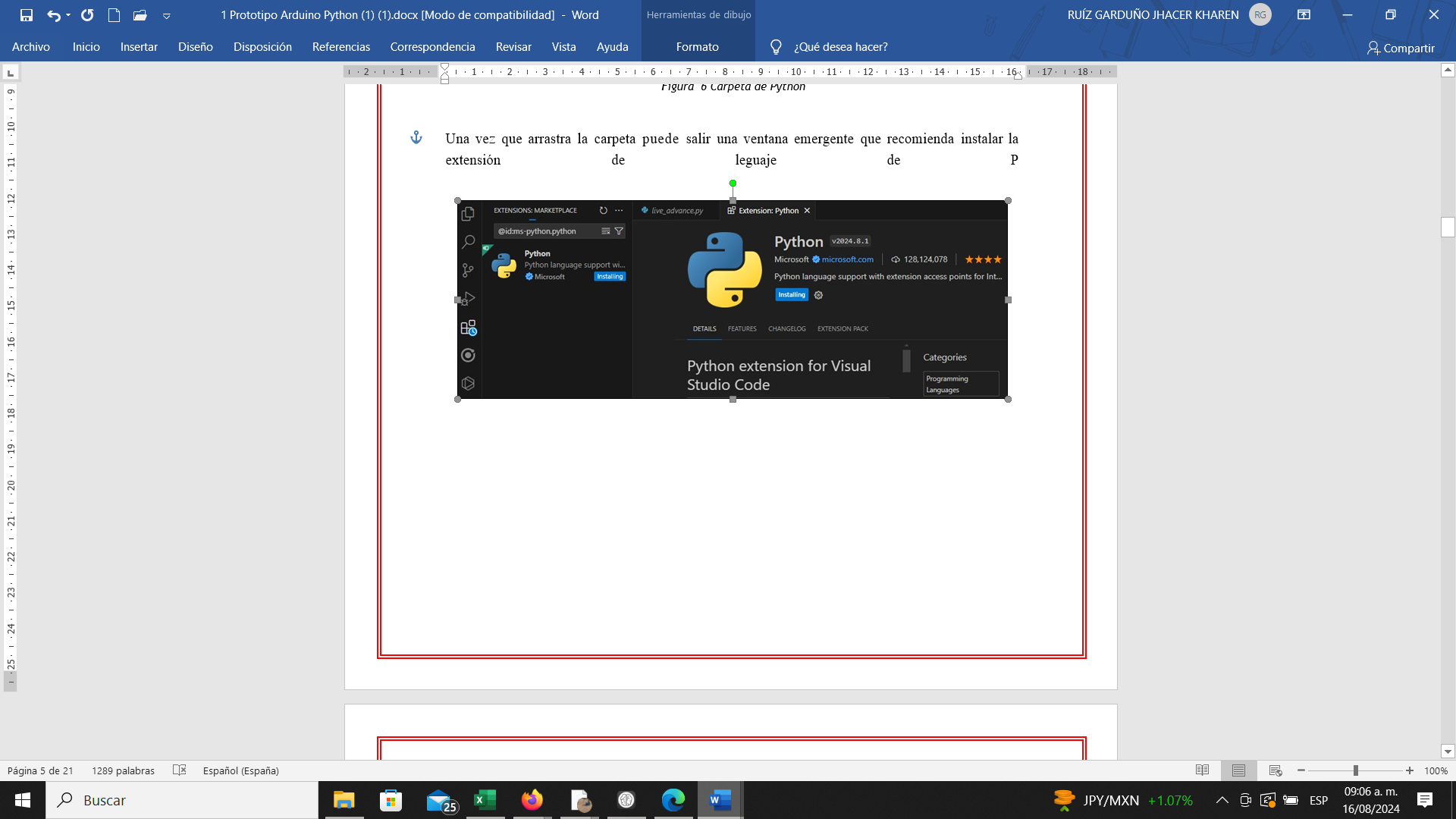


Figure 7 Extension Installation

Once the folder is in VSC, work only on the code named live\_advance.py where you will make the modifications.

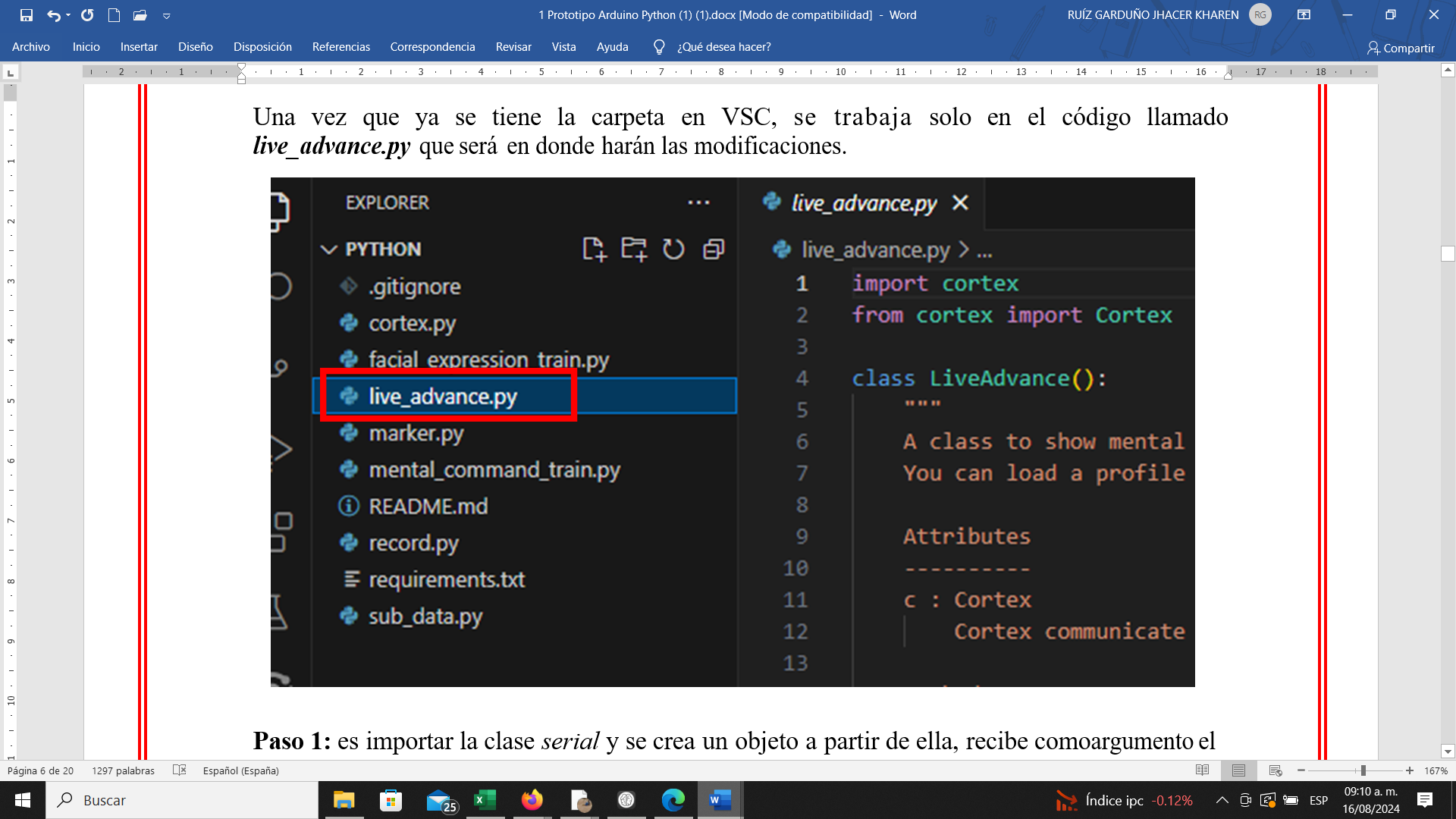


Figure 8 live\_advance Code

**Step 1:** Import the serial class and create an object from it, receiving the Arduino serial port and the baud rate as arguments.

**import serial**

**ser = serial.Serial('COM3',9600)**

## Imagen 18

Figure 9 Serial Class

It is crucial to use the port (in Figure 9’s example, “COM3”) where the Arduino is connected.

**NOTE:** This step can be skipped if you are not using or testing with Arduino yet.

**Step 2:** The second modification to the code is to add the following method (starting at line 221). Here, the action is obtained, and with the write method of the Serial class, the third letter of the action is sent to the serial port ‘COM3’ (previously defined). It’s important that the port matches the one to which Arduino is connected (If Arduino is detected as COM4, then COM4 should be used when creating the object).

**action = data.get('action', '')**

**ser.write(action[2].encode())**

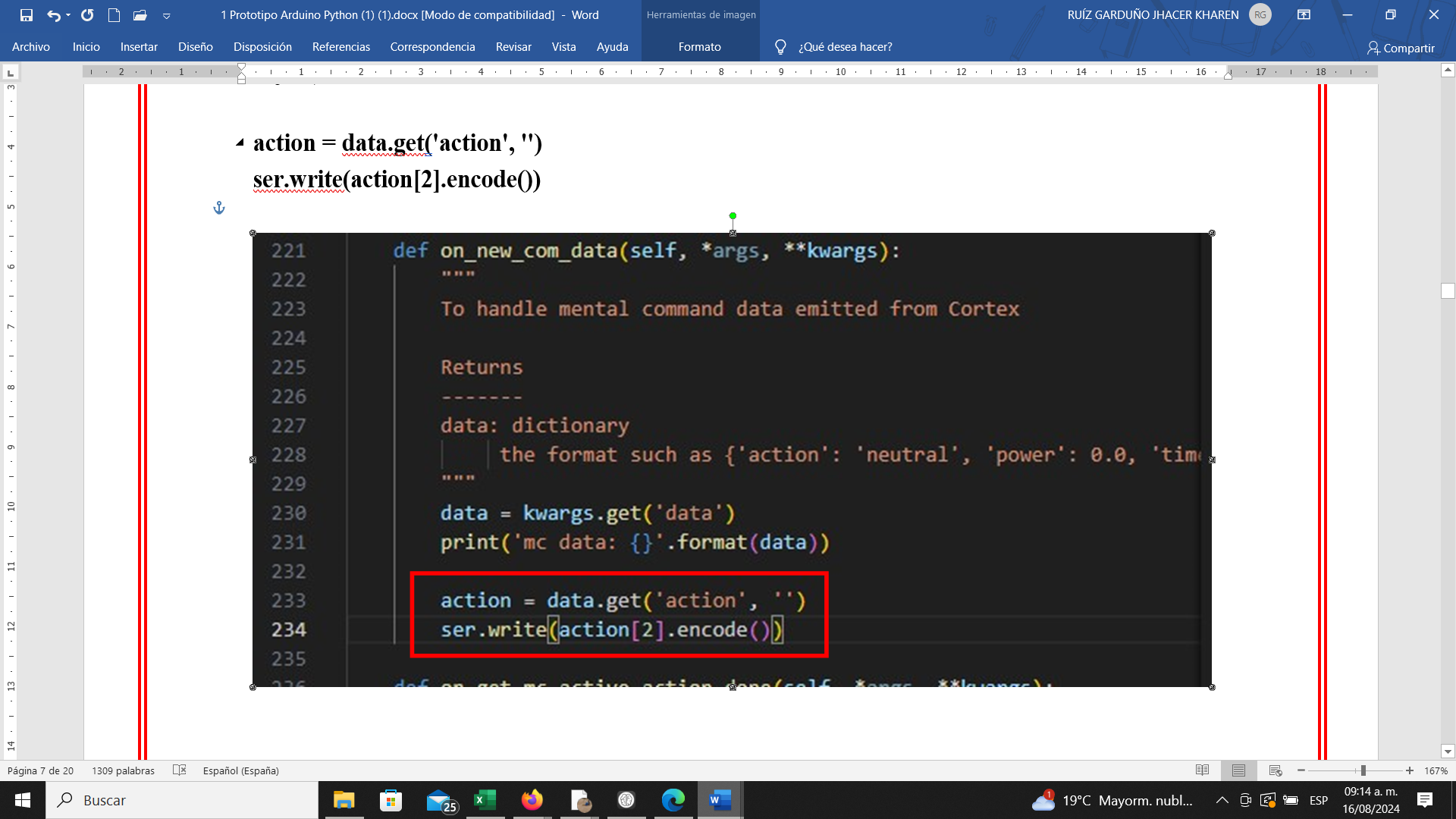


Figure 10 Write Method

Next, add both Client\_id and Client\_secret, which should already be created by this point. If not, refer to the updated EMOTIV Manual explaining this process.

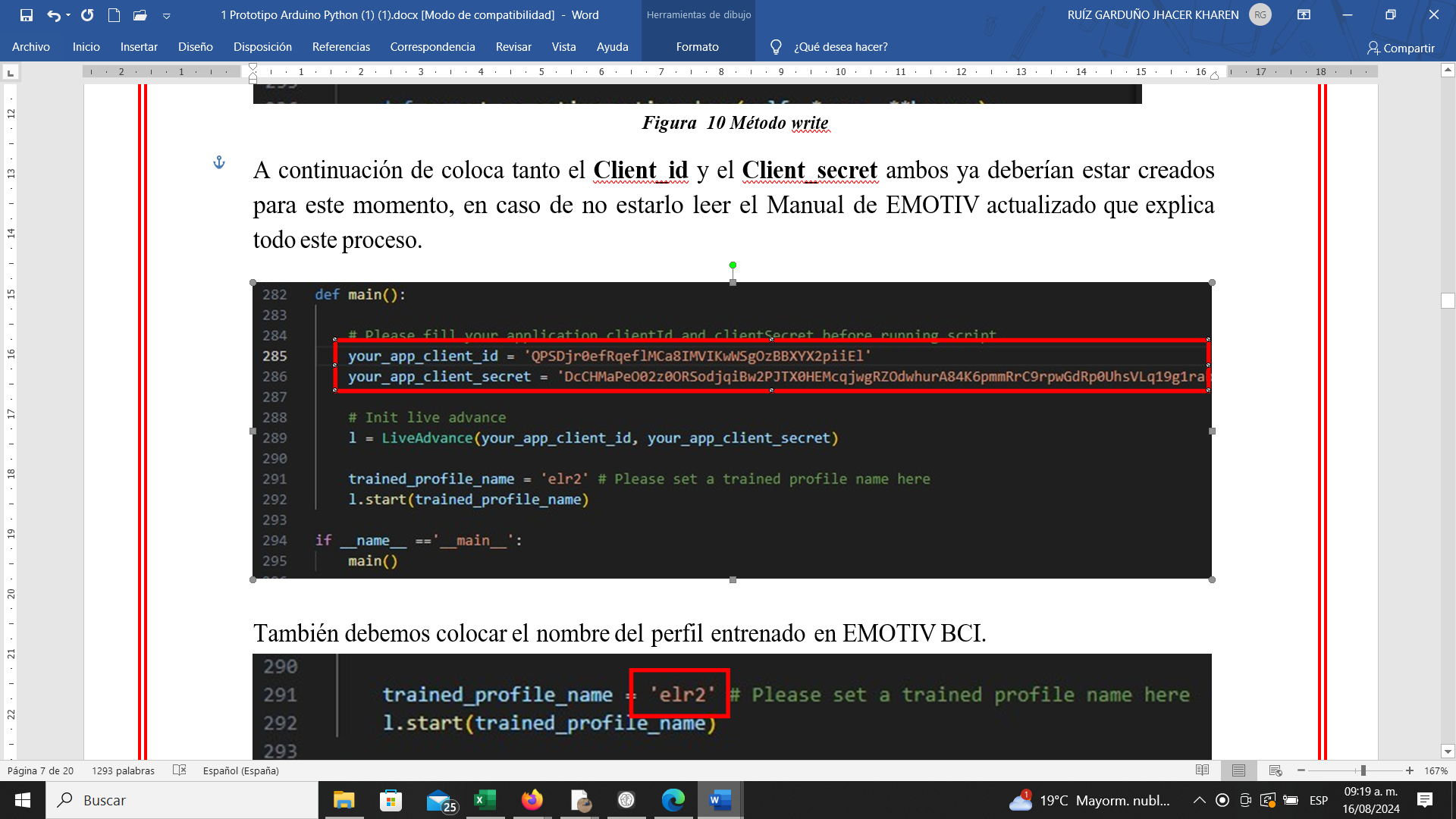


Figure 11 Access Keys

Also, specify the trained profile name in EMOTIV BCI.

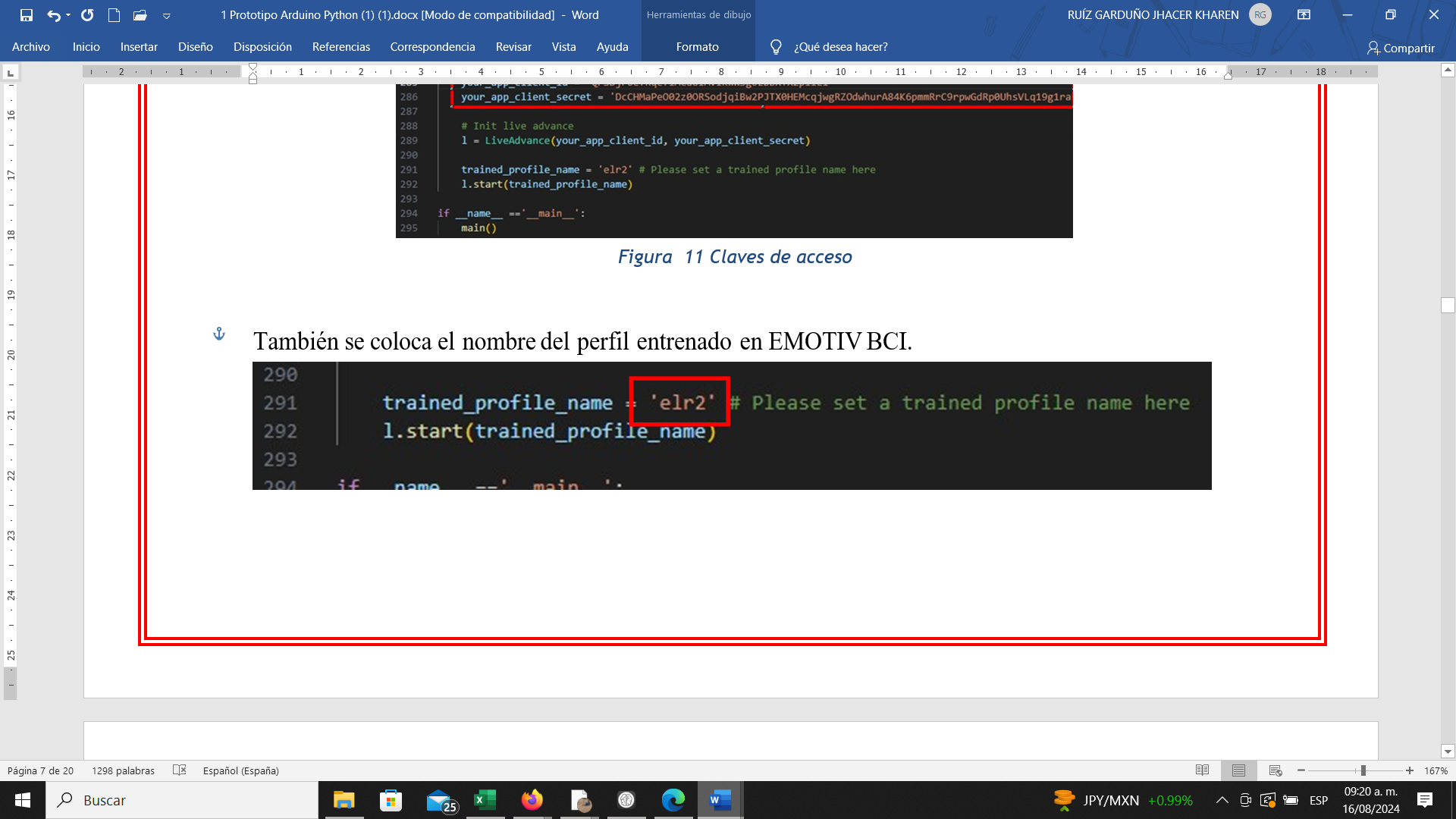


Figure 12 Trained Profile Name

Before doing this, you should have trained four directions for the same person using the EMOTIV BCI application. The following examples show how the profile name and trained movements appear.



Figure 13 Profile Movements

Enter the person’s name exactly as written in the application. Figure 14 shows the completion of the three items highlighted in orange.

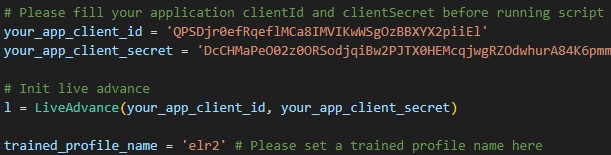


Figure 14 Client Name

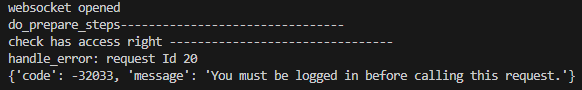
Before running the program, make sure you are logged into the EMOTIV LAUNCHER application, as the program will show an error otherwise.

Figure 15 Emotiv Launcher Error

**NOTE:** When running it for the first time, an alert message will appear in the EMOTIV LAUNCHER application that needs to be accepted to establish a connection between the code and the application.

When running the program again, if the headset is not connected, it will show:

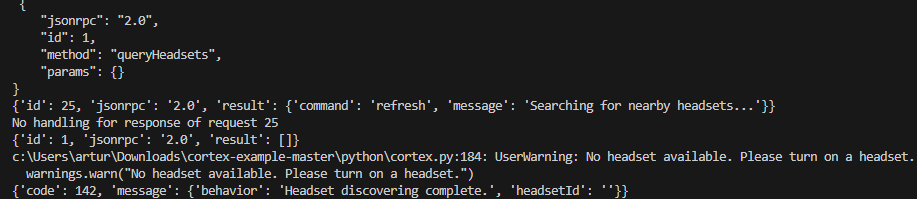


Figure 16 EMOTIV Headset Response

The solution is to connect the EMOTIV headset to the EMOTIV LAUNCHER and rerun the program in the upper right.

NOTE: Sensor configuration and concentration percentage must be done before running the program; otherwise, the program will show an error.

# Some execution test results.

## Imagen 36

Figure 17 Execution Tests

This process can be done with the physical headset or, if not available, configure and use a virtual headset.

# Arduino Instructions Manual (LEDs)

## Diagram.

## Imagen 37

Figure 18 Arduino Connection Diagram

## Arduino Code

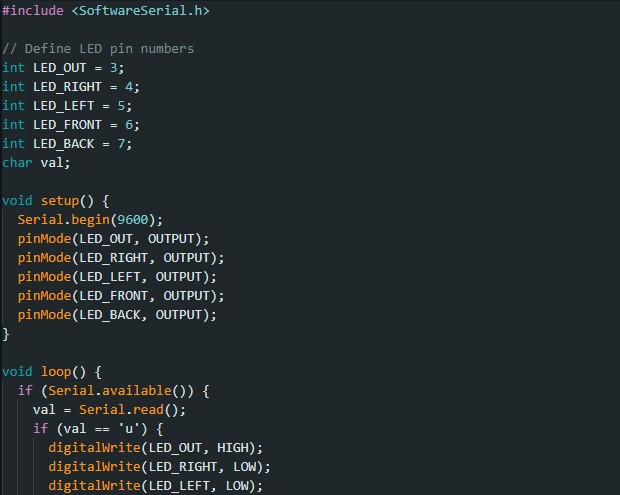


Figure 19 Arduino Code

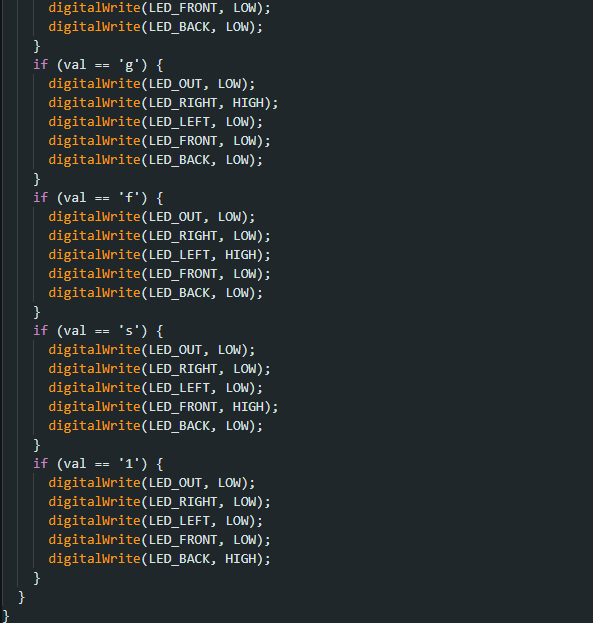


Figure 20 Variable Definitions

## Python Code

Import the serial class and create an object from it, receiving the Arduino serial port and baud rate as arguments.

**import serial**

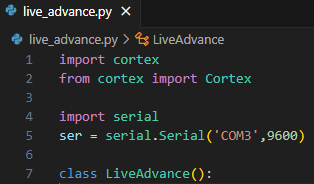
**ser = serial.Serial('COM3', 9600)**

Figure 21 Serial Port Definition

# Virtual Headset

Select "Add a Virtual Brainwear device."

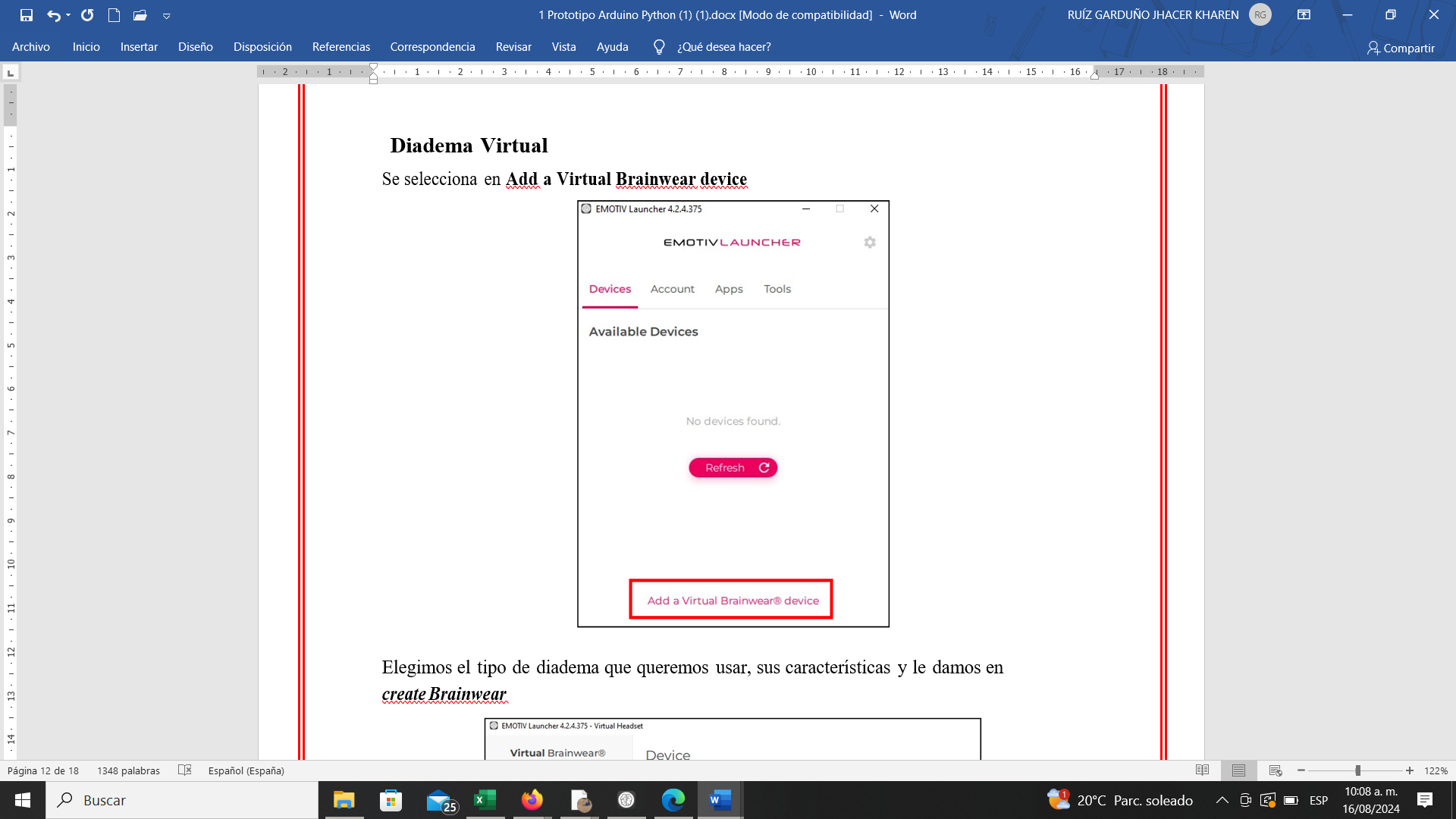


Figure 22 Add Headset

Choose the type of headset to use, its characteristics, and click "Create Brainwear.

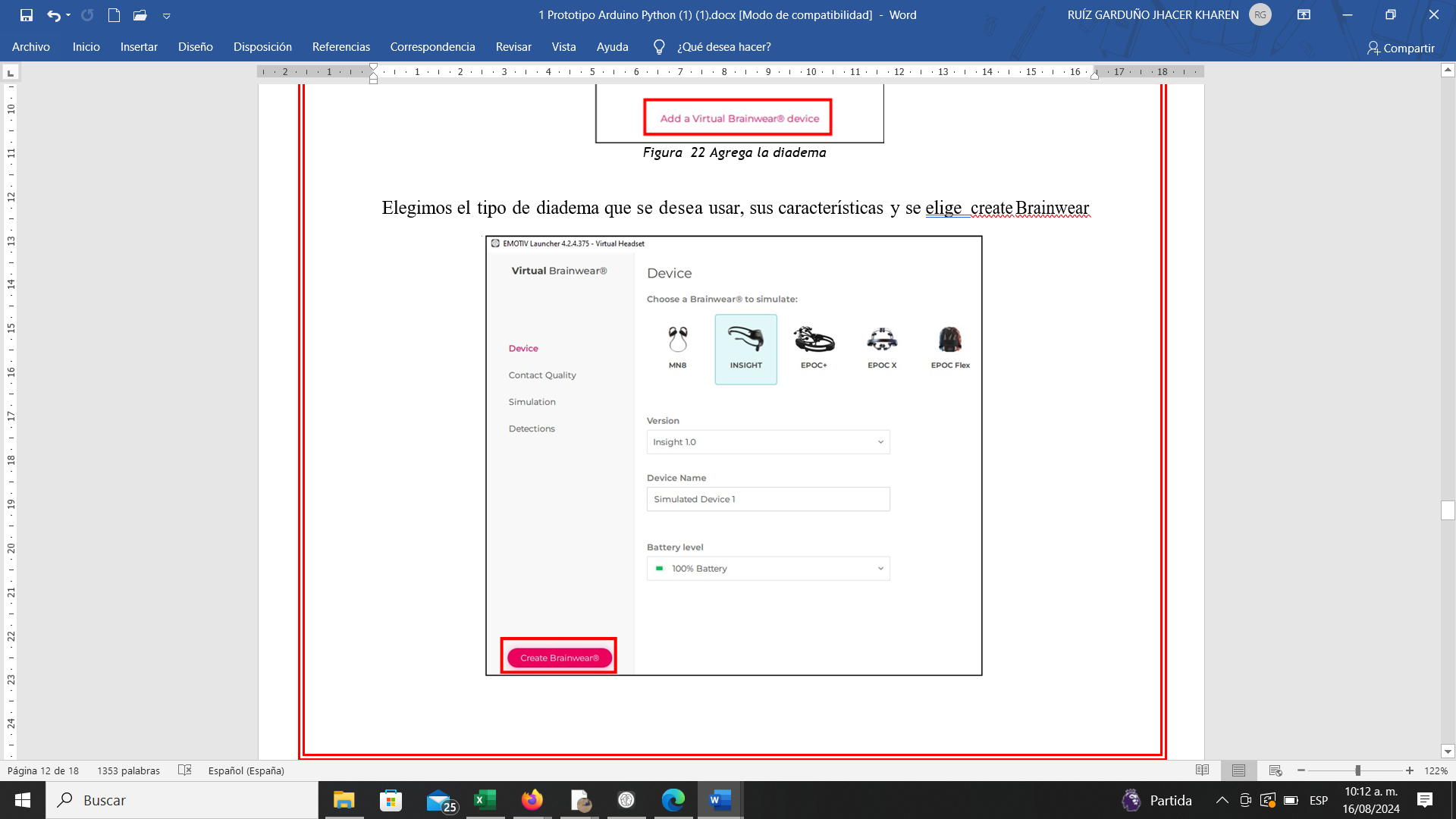


Figure 23 Headset Types

Once created, turn it on.

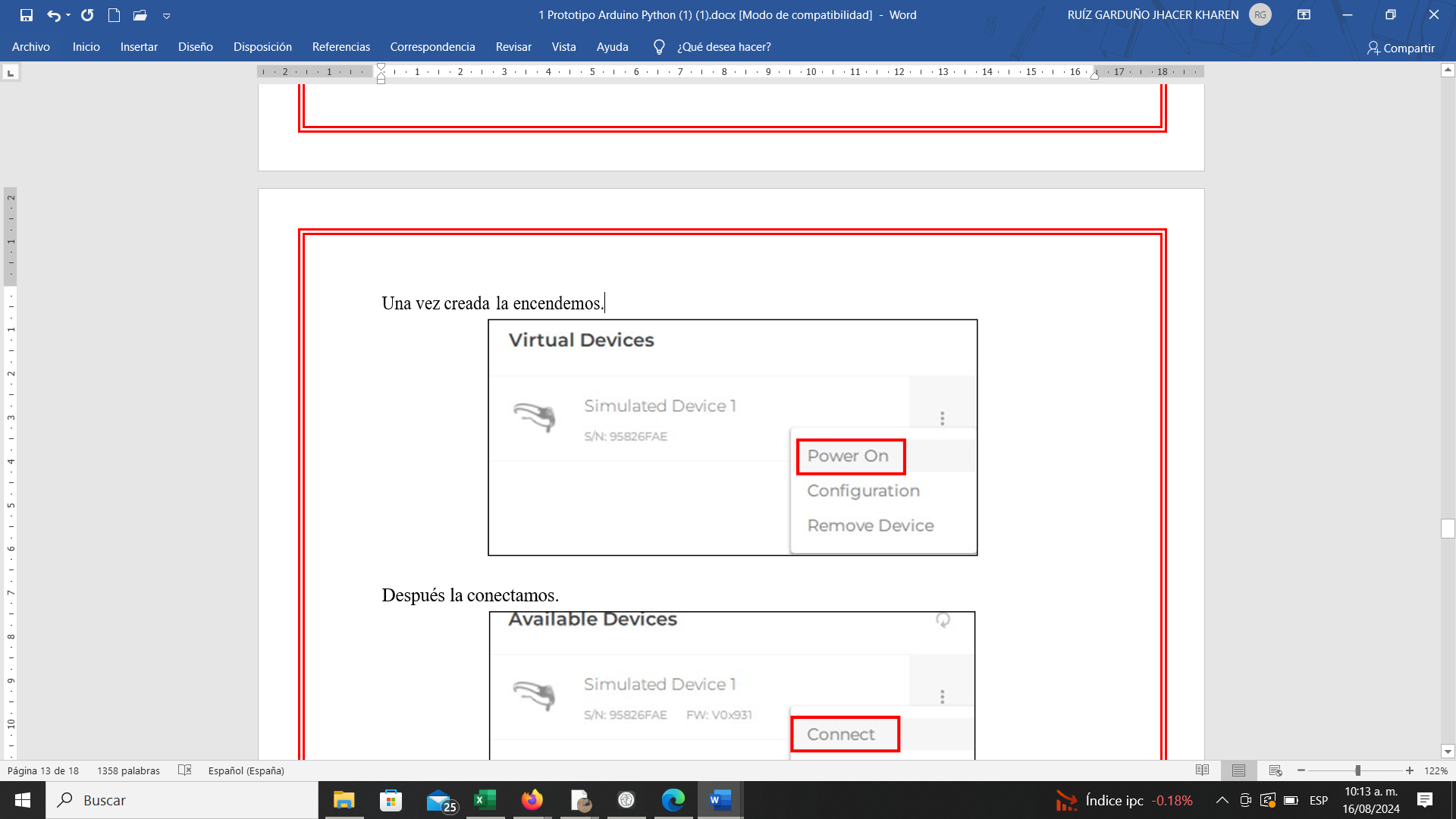


Figure 24 Headset On

Then connect it.

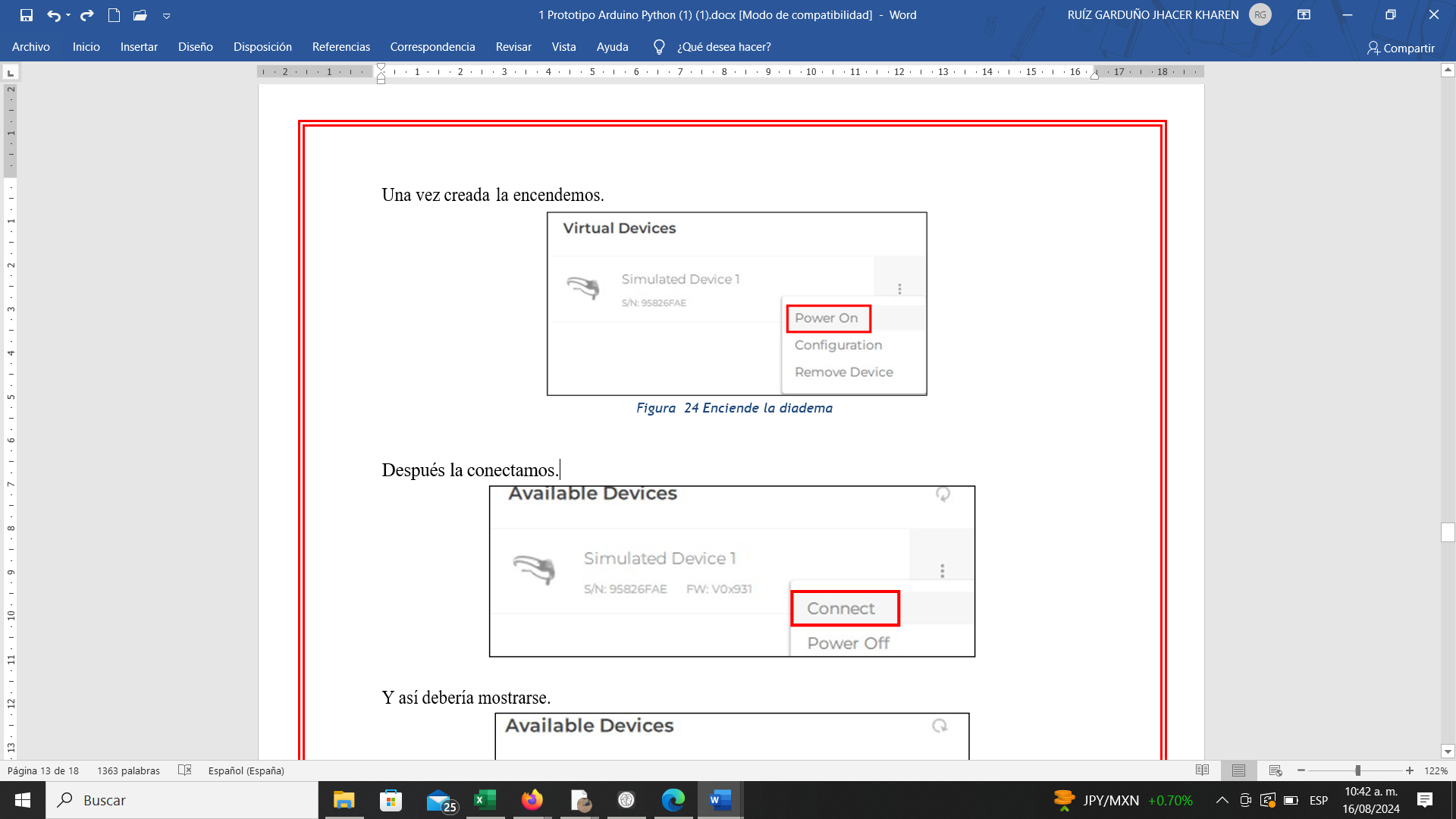


Figure 25 Headset Connection

It should appear as shown.

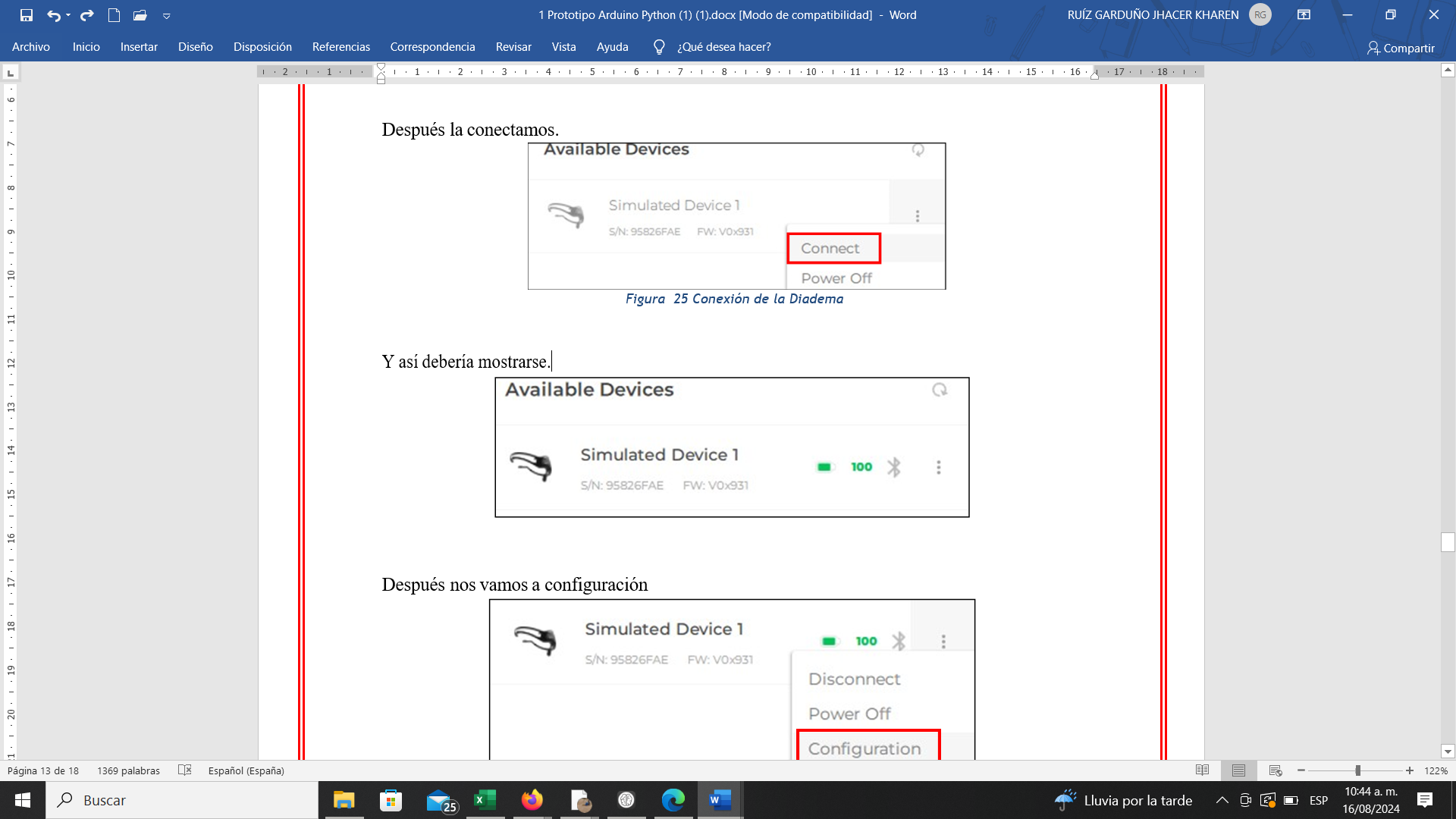


Figure 26 Connection Validation

Then go to configuration.

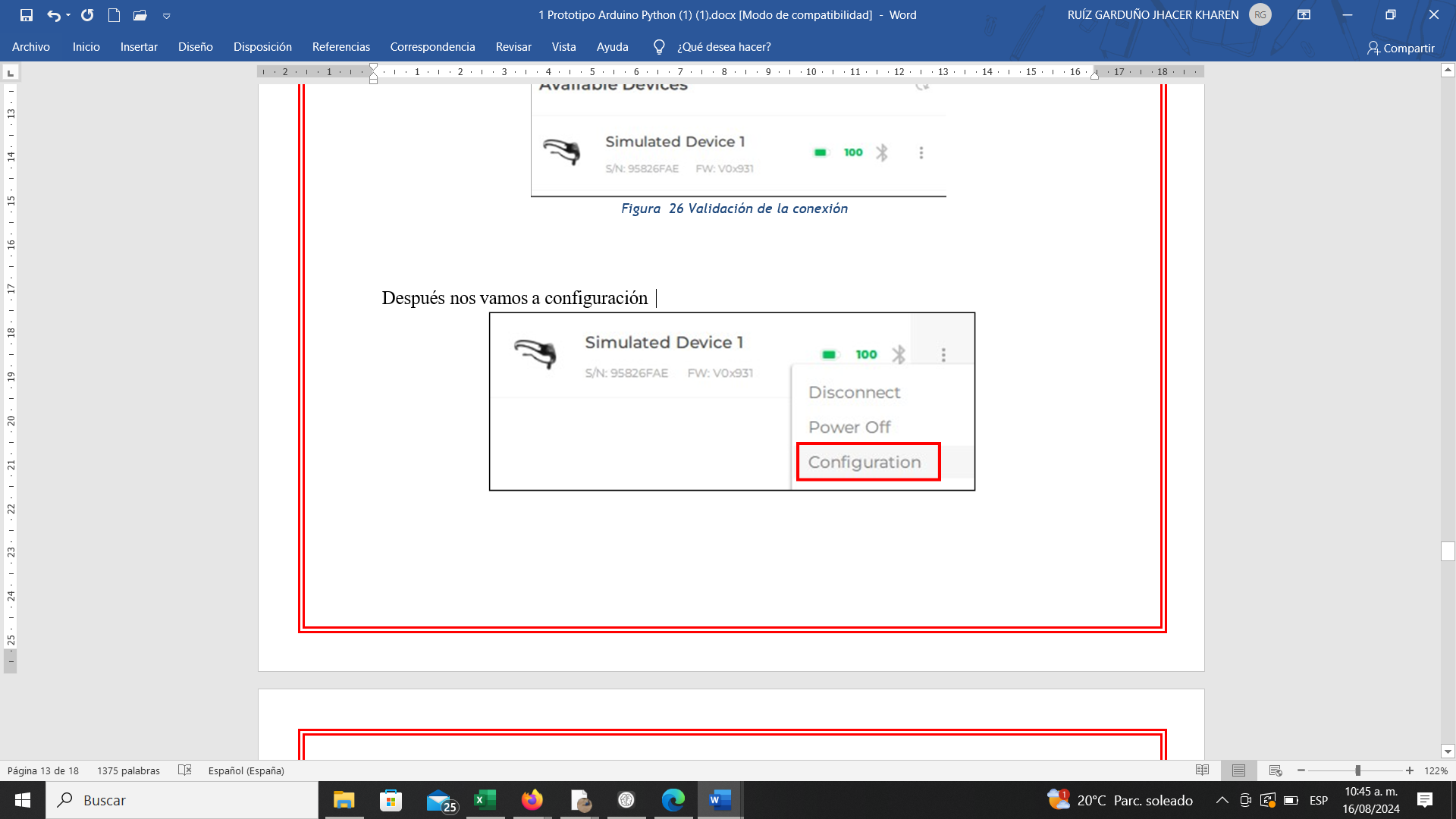


Figure 27 Headset Configuration

In the Detection and Mental Commands section, select the movement to be executed by the virtual headset for testing with VSC or with our Arduino prototype.

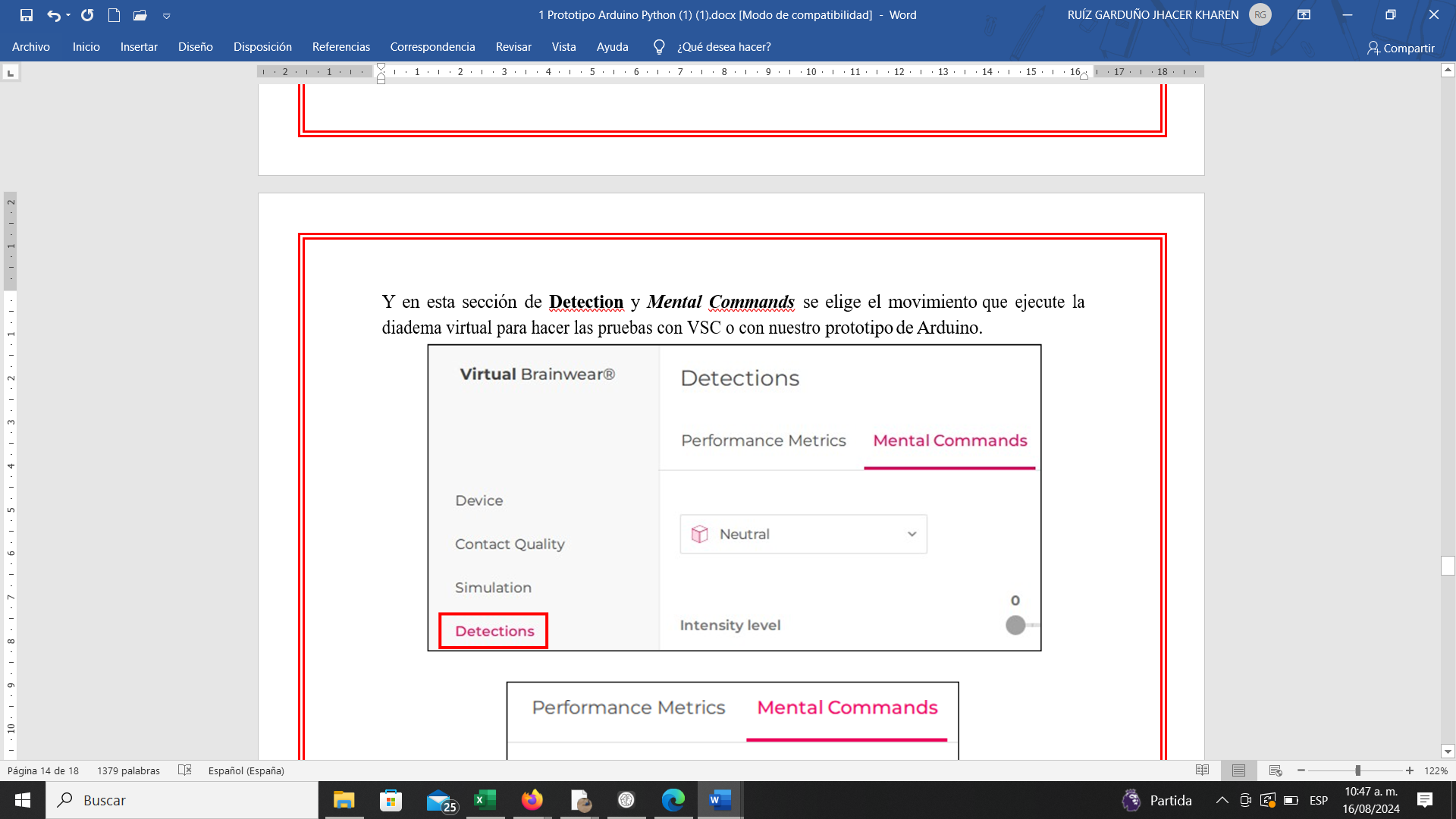


Figure 28 Movement Configuration

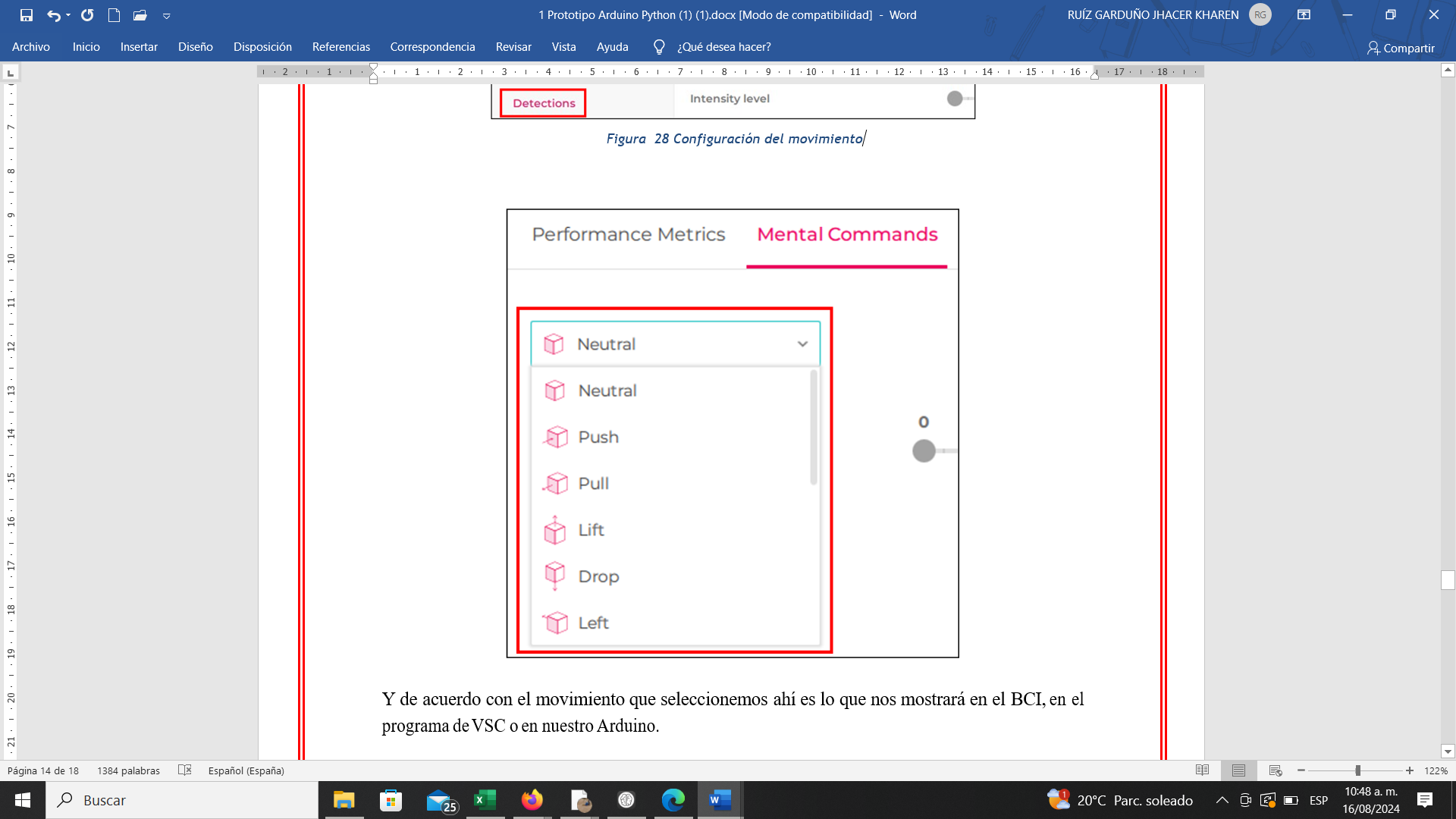


Figure 29 Movement Types

The selected movement will be displayed in the BCI, in the VSC program, or in our Arduino.

# Arduino Instructions Manual (Cart)

**Diagram**

## Imagen 48

Figure 30 Design Diagram

## Texto Descripción generada automáticamenteArduino Code

Figure 31 Arduino Configuration Code (Part 1)

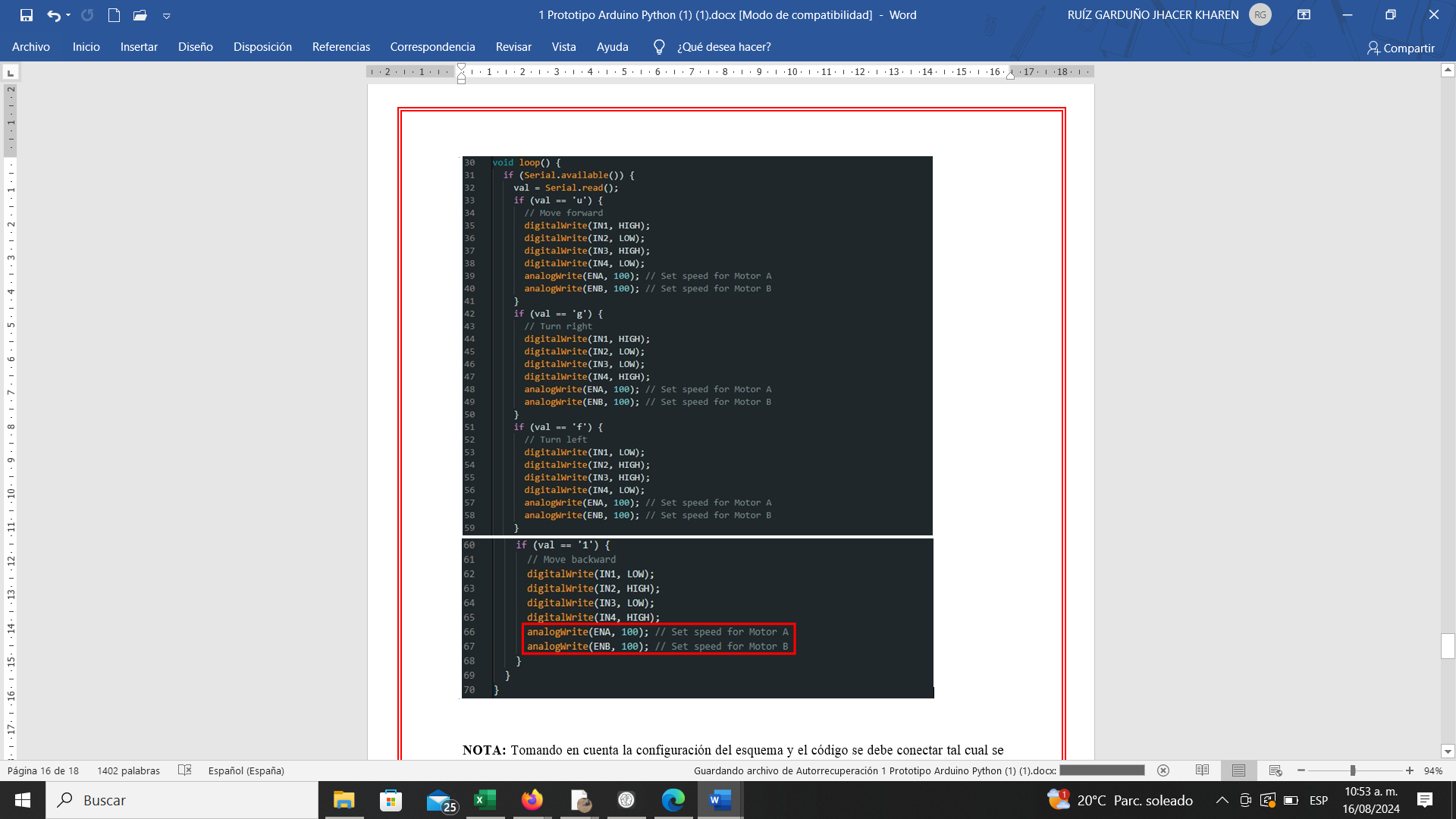


Figure 32 Arduino Configuration Code (Part 2)

**NOTE:** Based on the diagram and code configuration, connect as shown in Figure 32, which can be an Arduino UNO or NANO, always verifying the use of digital pins.

In the Arduino code, the variables ENA and ENB (highlighted in red) can have their values modified from 1 to 255, depending on the speed required for the motors (the higher the value, the faster the movement).

In the code example, values of 100 are used, a suitable speed to test the cart.

## Python Code Modifications

**Step 1:** Add the following code in line 221:

## action = data.get('action', '') if action == 'lift':

## ser.write('u'.encode()) # Subir -> Avanzar

## elif action == 'right':

## ser.write('g'.encode()) # Derecha -> Girar a la derecha elif action == 'left':

## ser.write('f'.encode()) # Izquierda -> Girar a la izquierda elif action == 'pull':

## ser.write('1'.encode()) # Jalar -> Retroceder

## Imagen 52

Figure 33 Variable Definition

The yellow-highlighted section is the block to identify, and the red box shows the code to be added, which reads the headset data and sends the movement command to Arduino.

**NOTE:** When copying and pasting the code into the method, it is crucial to respect the indentation of each function, as shown in the image; otherwise, it will produce a code error.

Similarly, this can be tested with the virtual headset before using the physical headset.

**Possible Errors**

When starting to install libraries, you may encounter the following error, which indicates that pip needs to be updated. The screen will show the command to use for updating.

## Texto Descripción generada automáticamentepython.exe -m pip install --upgrade pip

Figure 34 Screen Error