

**Prototipo Arduino Python**

**Inrtegrantes:**

Griselda Cortes Barrera

Claudia Teresa González Alfaro

Jhacer Kharen Ruiz Garduño

Jorge Abraham Jiménez Alfaro

**Contenido**

[Manual de instrucciones para Python 3](#_Toc174699141)

[Manual de instrucciones para Arduino (LED´s) 10](#_Toc174699153)

[Diadema Virtual 12](#_Toc174699160)

[Manual de instrucciones para Arduino (Carrito) 15](#_Toc174699161)

**Índice de figuras**

[Figura 1 Pip install websocket-client 3](file:///C:\Users\JLMJKR\Downloads\1%20Prototipo%20Arduino%20Python%20(1)%20(1).docx#_Toc174699239)

[Figura 2 Pip install Python-dipatch 3](file:///C:\Users\JLMJKR\Downloads\1%20Prototipo%20Arduino%20Python%20(1)%20(1).docx#_Toc174699240)

[Figura 3 Pip install cortex2 3](file:///C:\Users\JLMJKR\Downloads\1%20Prototipo%20Arduino%20Python%20(1)%20(1).docx#_Toc174699241)

[Figura 4 Pyhton -m pip install pyserial 4](file:///C:\Users\JLMJKR\Downloads\1%20Prototipo%20Arduino%20Python%20(1)%20(1).docx#_Toc174699242)

[Figura 5 Pyhton -m pip install serial 4](file:///C:\Users\JLMJKR\Downloads\1%20Prototipo%20Arduino%20Python%20(1)%20(1).docx#_Toc174699243)

[Figura 6 Carpeta de Python 5](#_Toc174699244)

[Figura 7 Instalación de la extensión 5](#_Toc174699245)

[Figura 8 Código live\_advance 6](#_Toc174699246)

[Figura 9 Clase serial 6](#_Toc174699247)

[Figura 10 Método write 7](#_Toc174699248)

[Figura 11 Claves de acceso 7](#_Toc174699249)

[Figura 12 Nombre del perfil del entrenamiento 7](#_Toc174699250)

[Figura 13 Perfil de los movimientos 8](#_Toc174699251)

[Figura 14 Nombre del cliente 8](#_Toc174699252)

[Figura 15 Error Emotiv Launcher 8](file:///C:\Users\JLMJKR\Downloads\1%20Prototipo%20Arduino%20Python%20(1)%20(1).docx#_Toc174699253)

[Figura 16 Respuesta de la Diadema EMOTIV 9](#_Toc174699254)

[Figura 17 Pruebas de ejecución 9](#_Toc174699255)

[Figura 18 Esquema de conexión con Arduino 10](#_Toc174699256)

[Figura 19 Código de Arduino 10](file:///C:\Users\JLMJKR\Downloads\1%20Prototipo%20Arduino%20Python%20(1)%20(1).docx#_Toc174699257)

[Figura 20 Definición de variables 11](#_Toc174699258)

[Figura 21 Definición del puerto serial 11](file:///C:\Users\JLMJKR\Downloads\1%20Prototipo%20Arduino%20Python%20(1)%20(1).docx#_Toc174699259)

[Figura 22 Agrega la diadema 12](#_Toc174699260)

[Figura 23 Tipos de diademas 12](#_Toc174699261)

[Figura 24 Enciende la diadema 13](#_Toc174699262)

[Figura 25 Conexión de la Diadema 13](#_Toc174699263)

[Figura 26 Validación de la conexión 13](#_Toc174699264)

[Figura 27Configuración de la Diadema 13](#_Toc174699265)

[Figura 28 Configuración del movimiento 14](#_Toc174699266)

[Figura 29 Tipos de movimiento 14](#_Toc174699267)

[Figura 30 Esquema del diseño 15](#_Toc174699268)

[Figura 31 Código de configuración del Arduino (parte 1) 15](file:///C:\Users\JLMJKR\Downloads\1%20Prototipo%20Arduino%20Python%20(1)%20(1).docx#_Toc174699269)

[Figura 32 Código de configuración del Arduino (parte 2) 16](#_Toc174699270)

[Figura 33 Definición de variables 17](#_Toc174699271)

[Figura 34 Error de pantalla 18](file:///C:\Users\JLMJKR\Downloads\1%20Prototipo%20Arduino%20Python%20(1)%20(1).docx#_Toc174699272)

# Manual de instrucciones para Python

En primera instancia se descarga las librerías que se utilizará para Python, que será el entorno en donde se ejecutará el programa con el que se conectará nuestro prototipo de Arduino.

Se revisa que se cuente con la versión más reciente de Pyhton, si no se tiene la se puede descargar del siguiente enlace: <https://www.python.org/downloads/>

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

Para comenzar, desde el CMD (de preferencia se ejecuta el CMD como administrador), y se ejecuta los siguientes comandos:

## pip install websocket-client

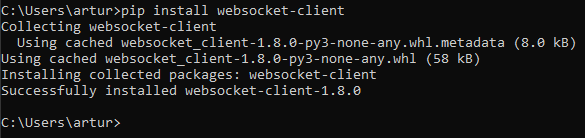


Figura 1 Pip install websocket-client

## pip install Python-dipatch

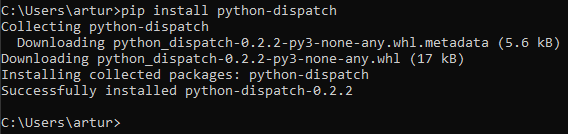


Figura 2 Pip install Python-dipatch

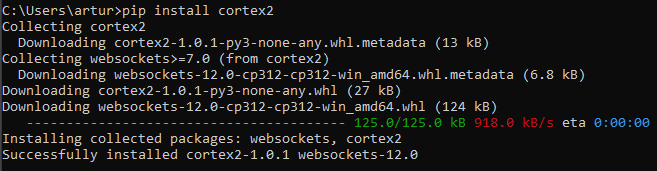
1. **pip install cortex2** (también instalar ***pip install cortex***, en caso de que no corra el programa)

Figura 3 Pip install cortex2

## pyhton -m pip install pyserial

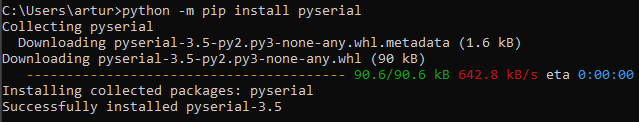


Figura 4 Pyhton -m pip install pyserial

## pyhton -m pip install serial

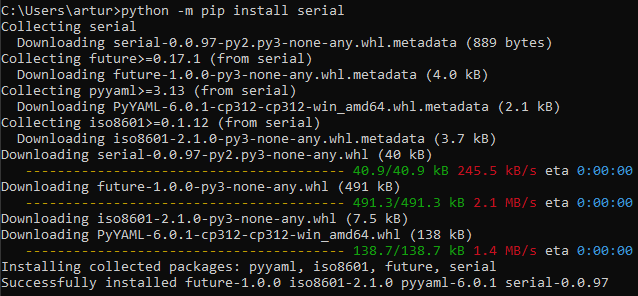


Figura 5 Pyhton -m pip install serial

Todas las instalaciones deben decir “Sucessfully istalled” lo que indica que las librerías se instalaron correctamente y se puede avanzar con el procedimiento.

Una vez que se cuenta con lo necesario en las librerías de Python, se procede a modificar el código que se encuentra en la carpeta llamada **Pyhton** dentro de la carpeta principal llamada ***cortex-example-master***

Se utiliza Visual Studio Code, se arrastra y suelta la carpeta **Python** al editor para abrirla directamente. O se puede abrir la carpeta desde *File*.

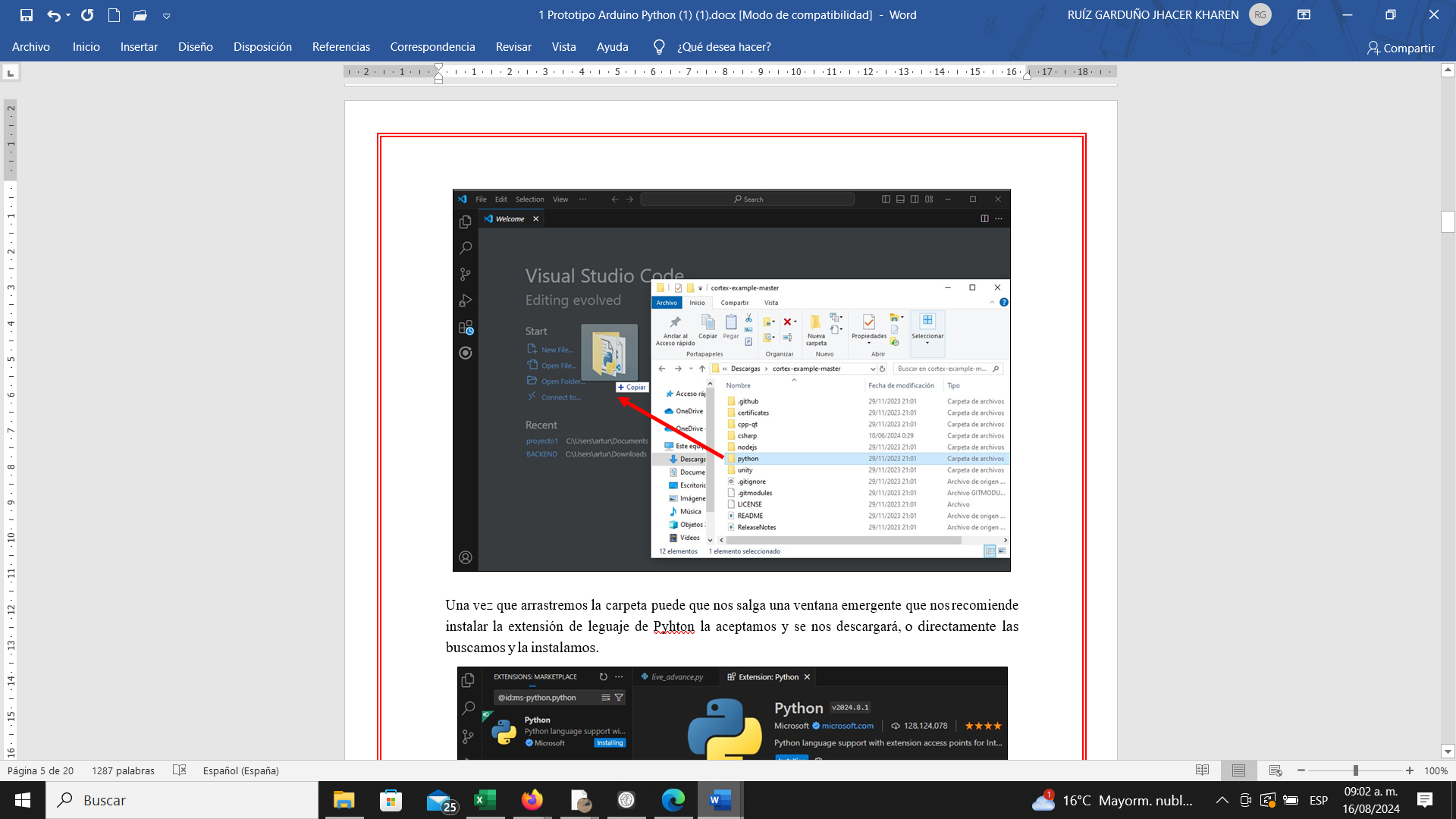


Figura 6 Carpeta de Python

Una vez que arrastra la carpeta puede salir una ventana emergente que recomienda instalar la extensión de leguaje de Python se acepta y se descargará , o directamente se busca y se instala.

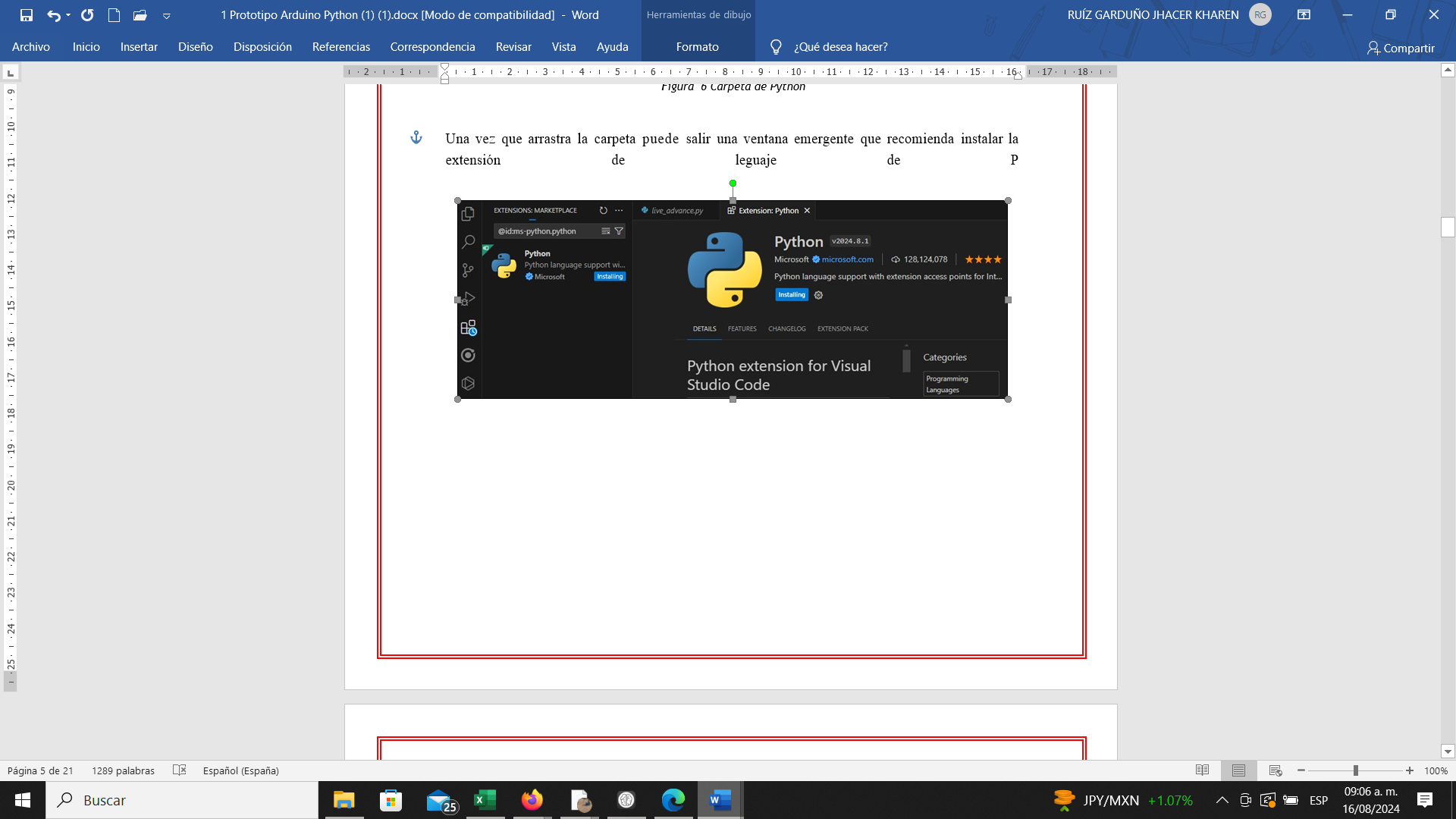


Figura 7 Instalación de la extensión

Una vez que ya se tiene la carpeta en VSC, se trabaja solo en el código llamado ***live\_advance.py*** que será en donde harán las modificaciones.

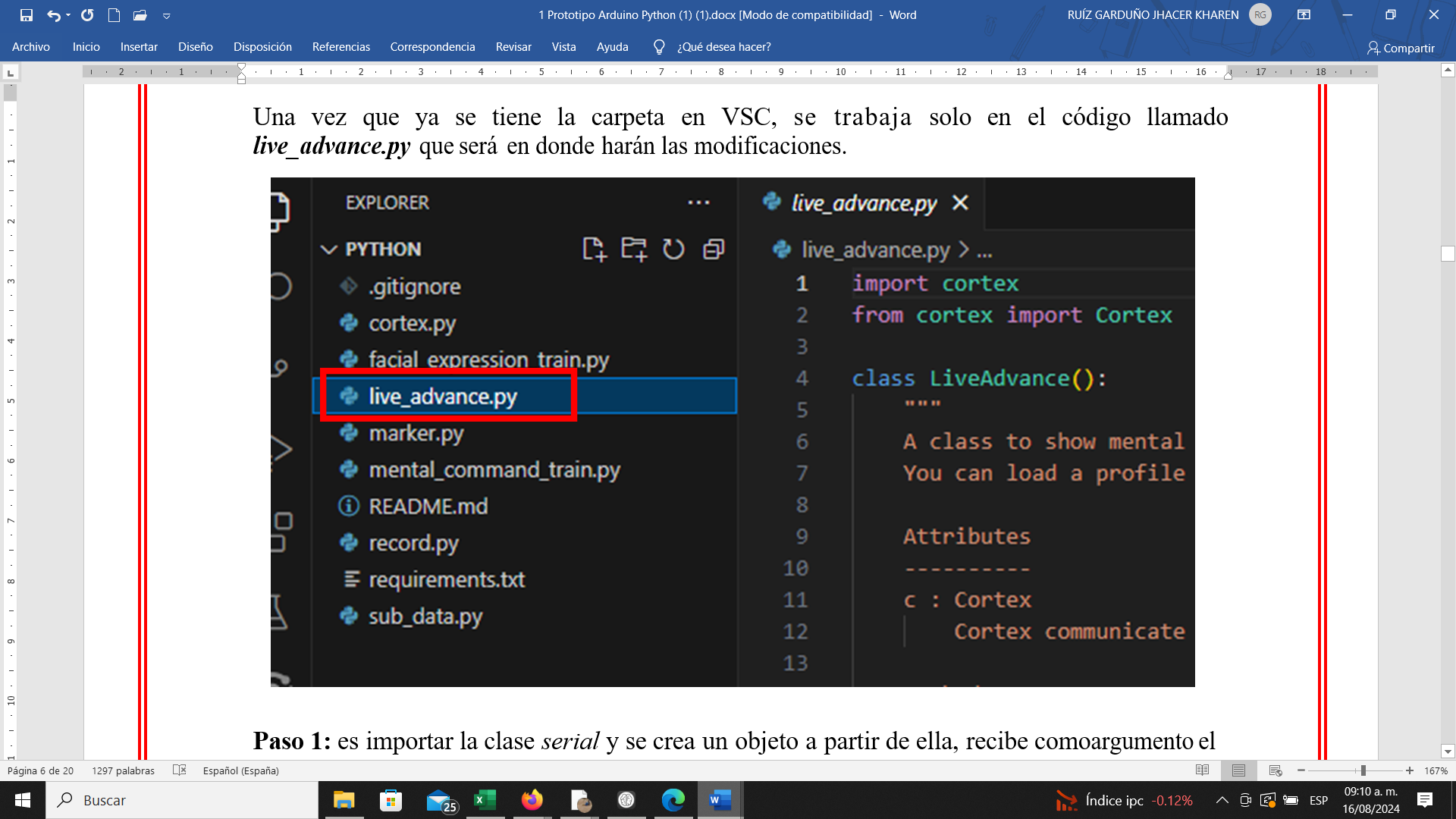


Figura 8 Código live\_advance

**Paso 1:** Es importar la clase *serial* y se crea un objeto a partir de ella, recibe como argumento el puerto serial del Arduino y los bauds en los que recibe los datos.

## import serial

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

## 

Figura 9 Clase serial

Es muy importante que el puerto (en el ejemplo de la figura 9 “COM3”) se coloque el puerto en el que está conectado el Arduino.

**NOTA:** Este paso se puede omitir si no se ocupará o probará con Arduino todavía.

**Paso 2:** La segunda modificación al código es agregar a este método (a partir de la línea 221), aquí se obtiene la acción y con el método *write* de la clase Serial se manda la tercera letra de la acción al puerto serial ‘COM3’ (anteriormente definido), cabe recalcar que el puerto debe ser el mismo al que el Arduino esté conectado (Si el Arduino se detecta como COM4, entonces se define COM4 al crear el objeto).

## action = data.get('action', '') ser.write(action[2].encode())

## 

Figura 10 Método write

A continuación de coloca tanto el **Client\_id** y el **Client\_secret** ambos ya deberían estar creados para este momento, en caso de no estarlo leer el Manual de EMOTIV actualizado que explica todo este proceso.

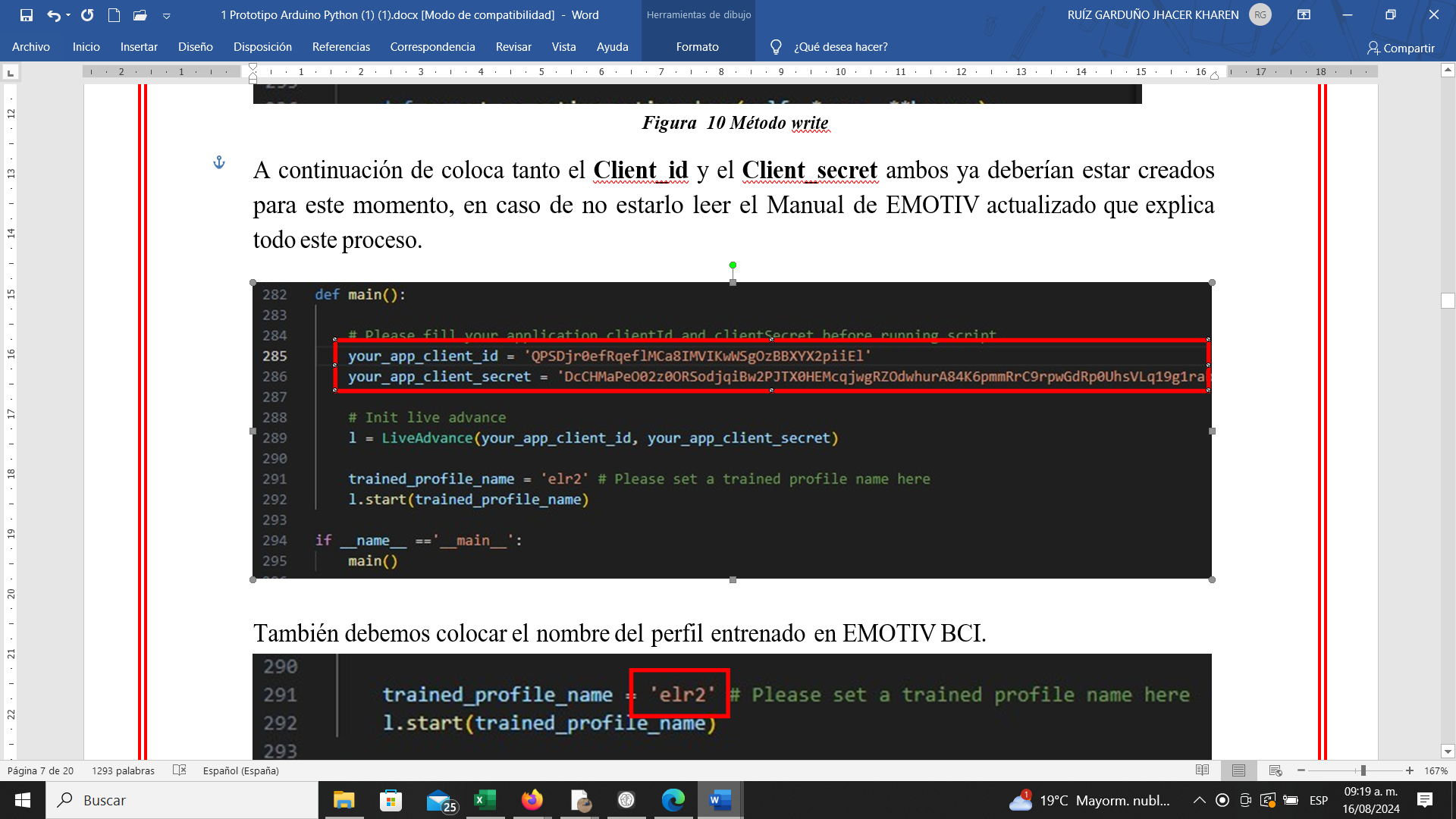


Figura 11 Claves de acceso

También se coloca el nombre del perfil entrenado en EMOTIV BCI.

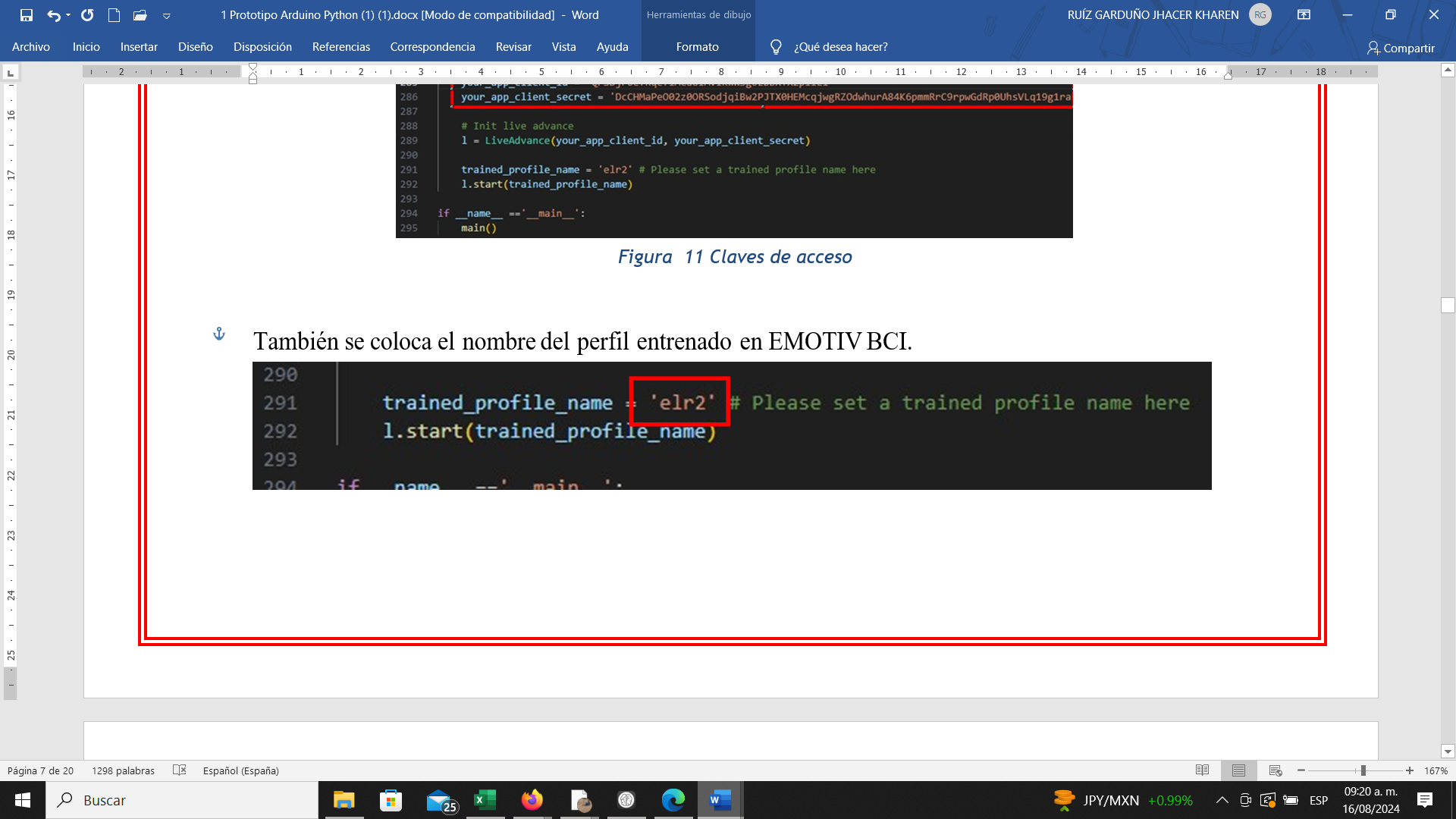


Figura 12 Nombre del perfil del entrenamiento

De igual manera antes de realizar esto se debió entrenar mediante la aplicación EMOTIV BCI las cuatro direcciones pedidas para una misma persona. Los siguientes ejemplos muestran como aparece el nombre del perfil y los movimientos entrenados.



Figura 13 Perfil de los movimientos

Se debe colocar el nombre de la persona tal cual como está escrito en la aplicación. En la imagen 14 se muestra el llenado de las 3 cosas estas se identifican de color naranja:

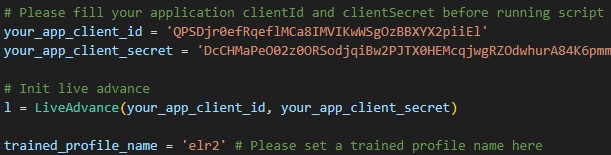


Figura 14 Nombre del cliente

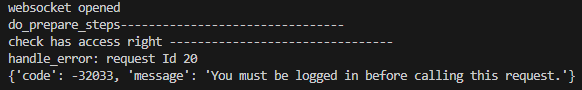
Antes de correr el programa se pide estar logeado en la aplicación de EMOTIV LAUNCHER ya que de no estarlo el programa marcara un error como se explica a continuación.

Figura 15 Error Emotiv Launcher

NOTA: al correrlo por primera vez deberá salir un mensaje de advertencia dentro de la aplicación EMOTIV LAUNCHER donde se debe aceptar para establecer la conexión entre código y aplicación.

Al volver a correr el programa en caso de **no tener la diadema** mandara esto:

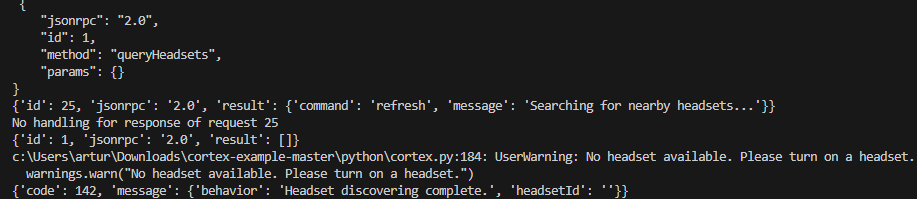


Figura 16 Respuesta de la Diadema EMOTIV

La solución simplemente es conectar la diadema correspondiente a EMOTIV LAUNCHER y volver a correr el programa en la parte superior derecha.

NOTA: Se debe realizar el proceso de configuración de sensores y porcentaje de concentración antes de correr el programa esto debido a que si no se realiza el programa mandara un error.

## Algunas pruebas de ejecución del programa.

## 

Figura 17 Pruebas de ejecución

Todo este proceso se puede hacer con la diadema física o en caso de no contar con ella, configurar y usar una diadema virtual

# Manual de instrucciones para Arduino (LED´s)

## Esquema.

## 

Figura 18 Esquema de conexión con Arduino

## Código Arduino

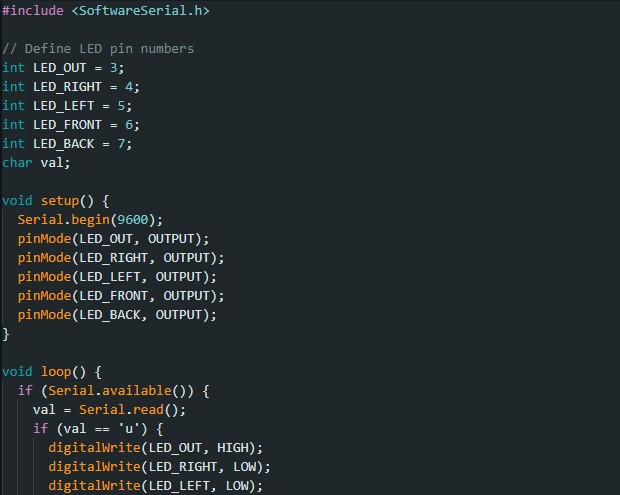


Figura 19 Código de Arduino

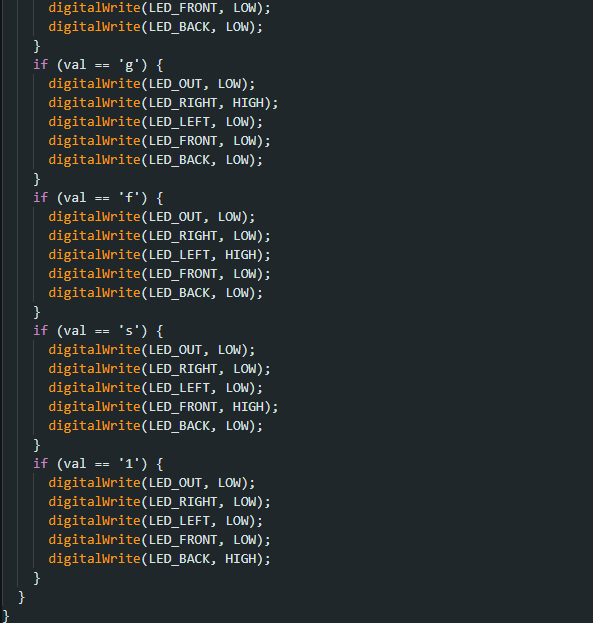


Figura 20 Definición de variables

## Código Python

Se imparte la clase serial y se genera un objeto a partir de ella, recibe como argumento el puerto serial del Arduino y los Bauds en los que recibe los datos.

## import serial

## Pantalla de computadora con letras Descripción generada automáticamente con confianza mediaser = serial.Serial('COM3', 9600)

Figura 21 Definición del puerto serial

# Diadema Virtual

Se selecciona en **Add a Virtual Brainwear device**

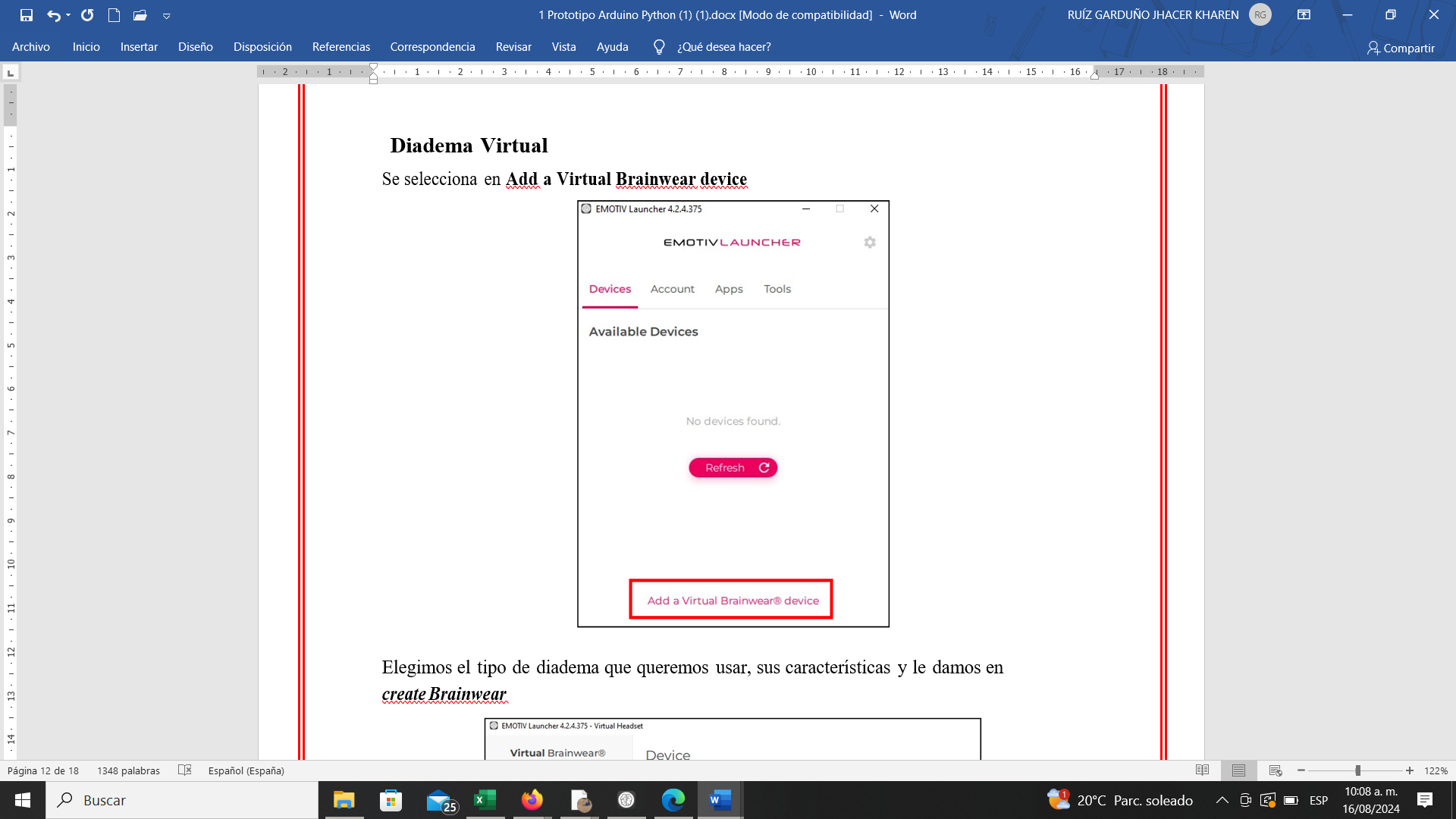


Figura 22 Agrega la diadema

Elegimos el tipo de diadema que se desea usar, sus características y se elige create Brainwear

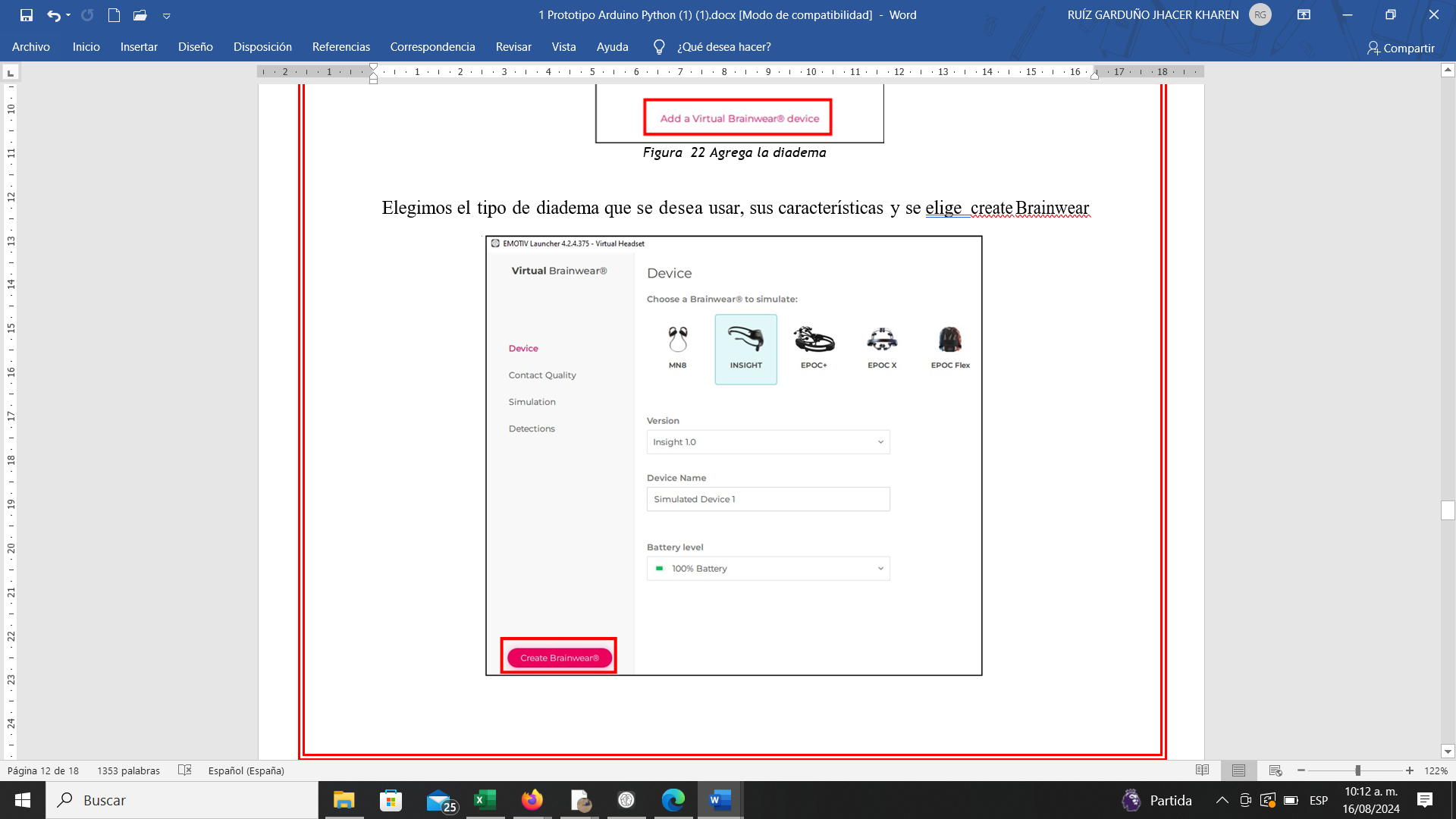


Figura 23 Tipos de diademas

Una vez creada la encendemos.

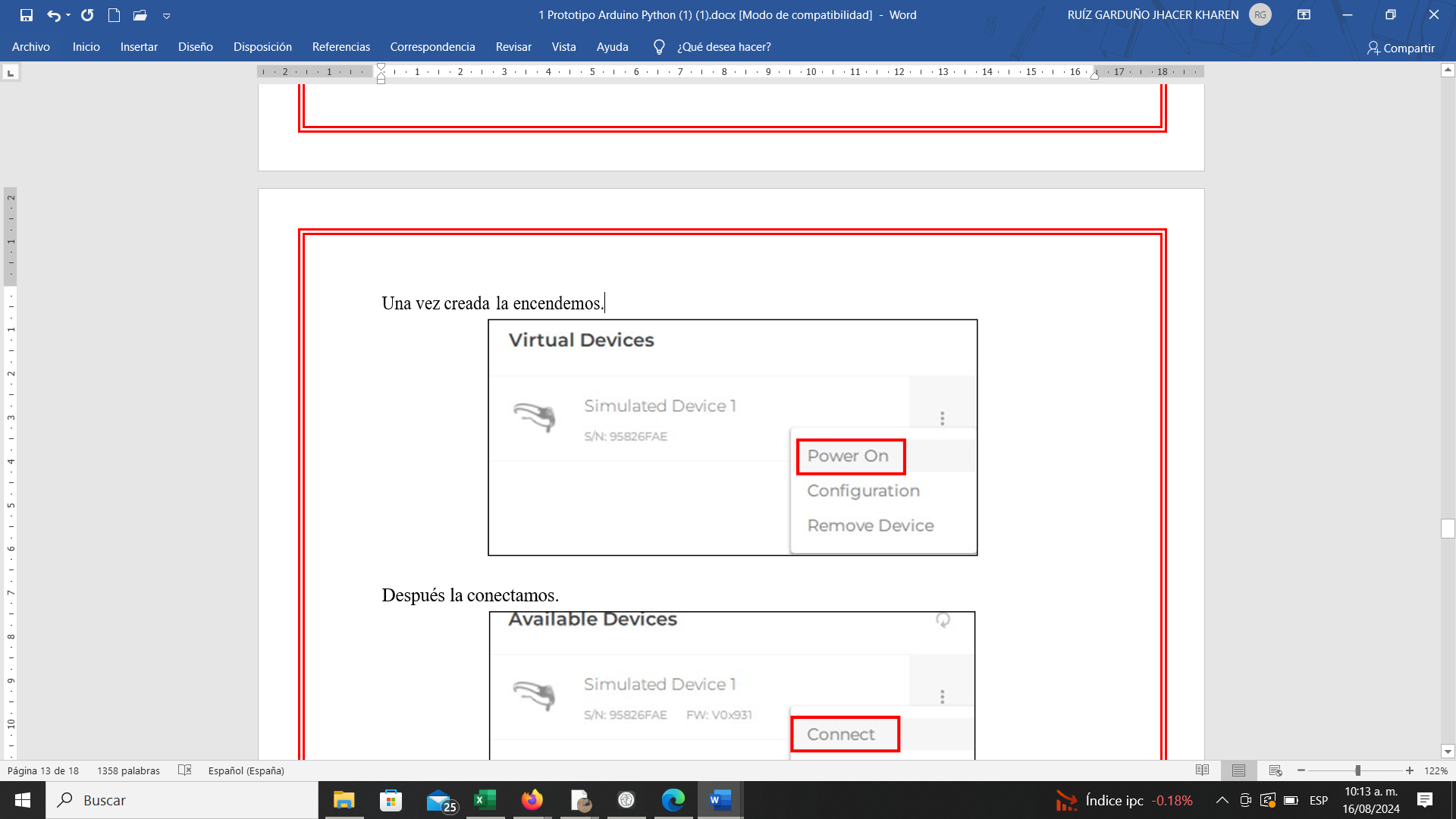


Figura 24 Enciende la diadema

Después la conectamos.

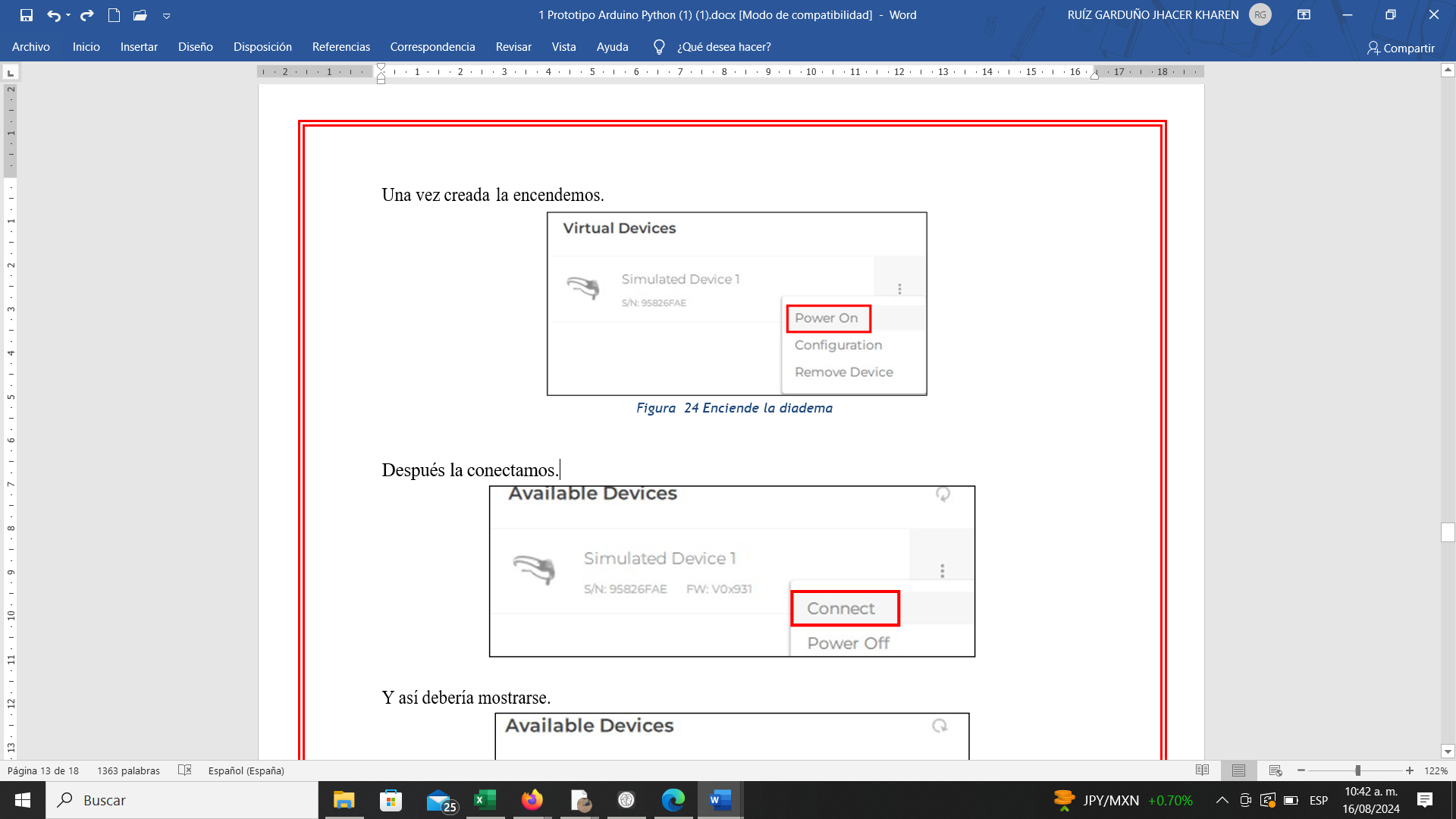


Figura 25 Conexión de la Diadema

Y así debería mostrarse.

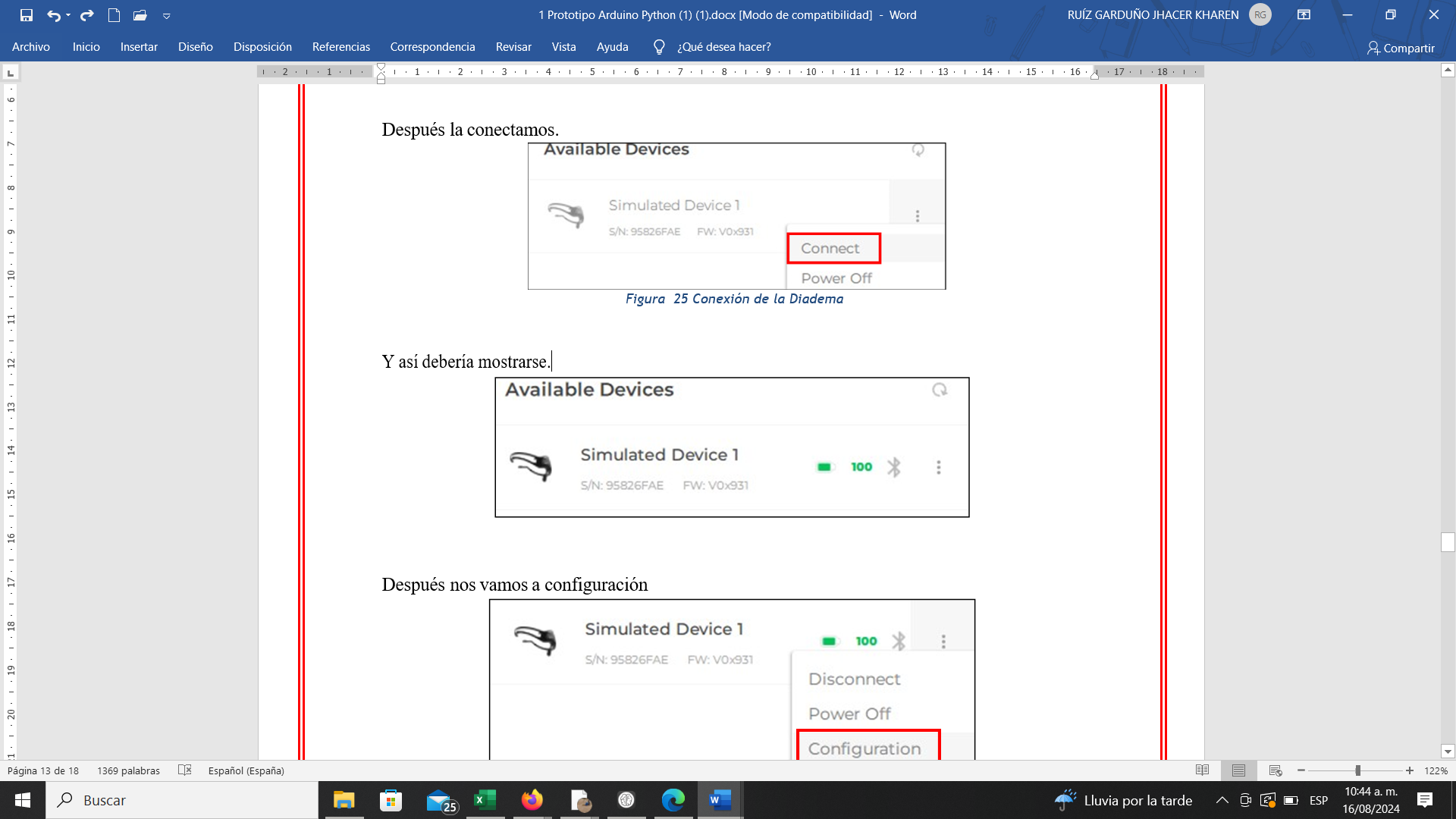


Figura 26 Validación de la conexión

Después nos vamos a configuración

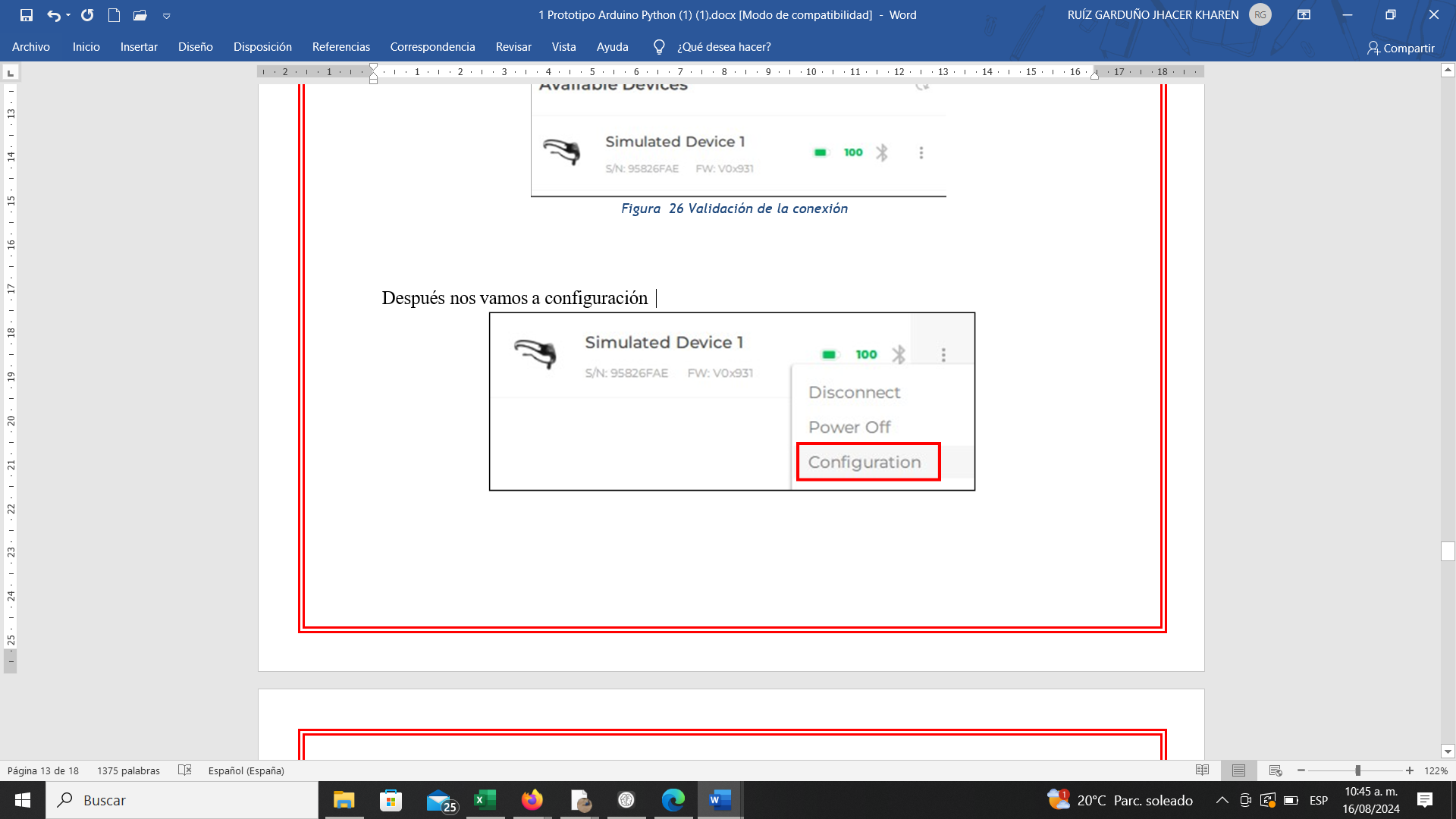


Figura 27Configuración de la Diadema

Y en esta sección de **Detection** y ***Mental Commands*** se elige el movimiento que ejecute la diadema virtual para hacer las pruebas con VSC o con nuestro prototipo de Arduino.

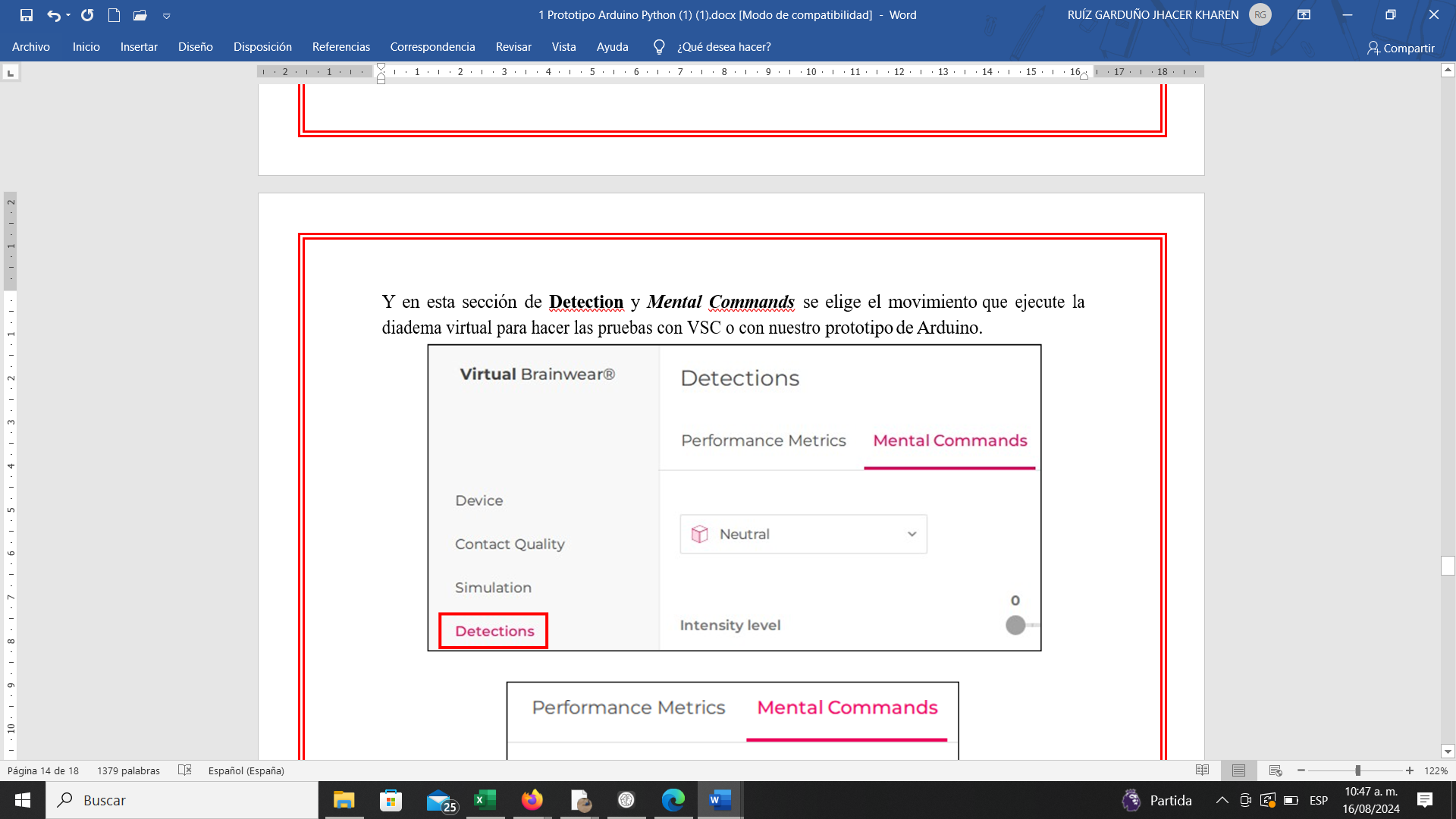


Figura 28 Configuración del movimiento

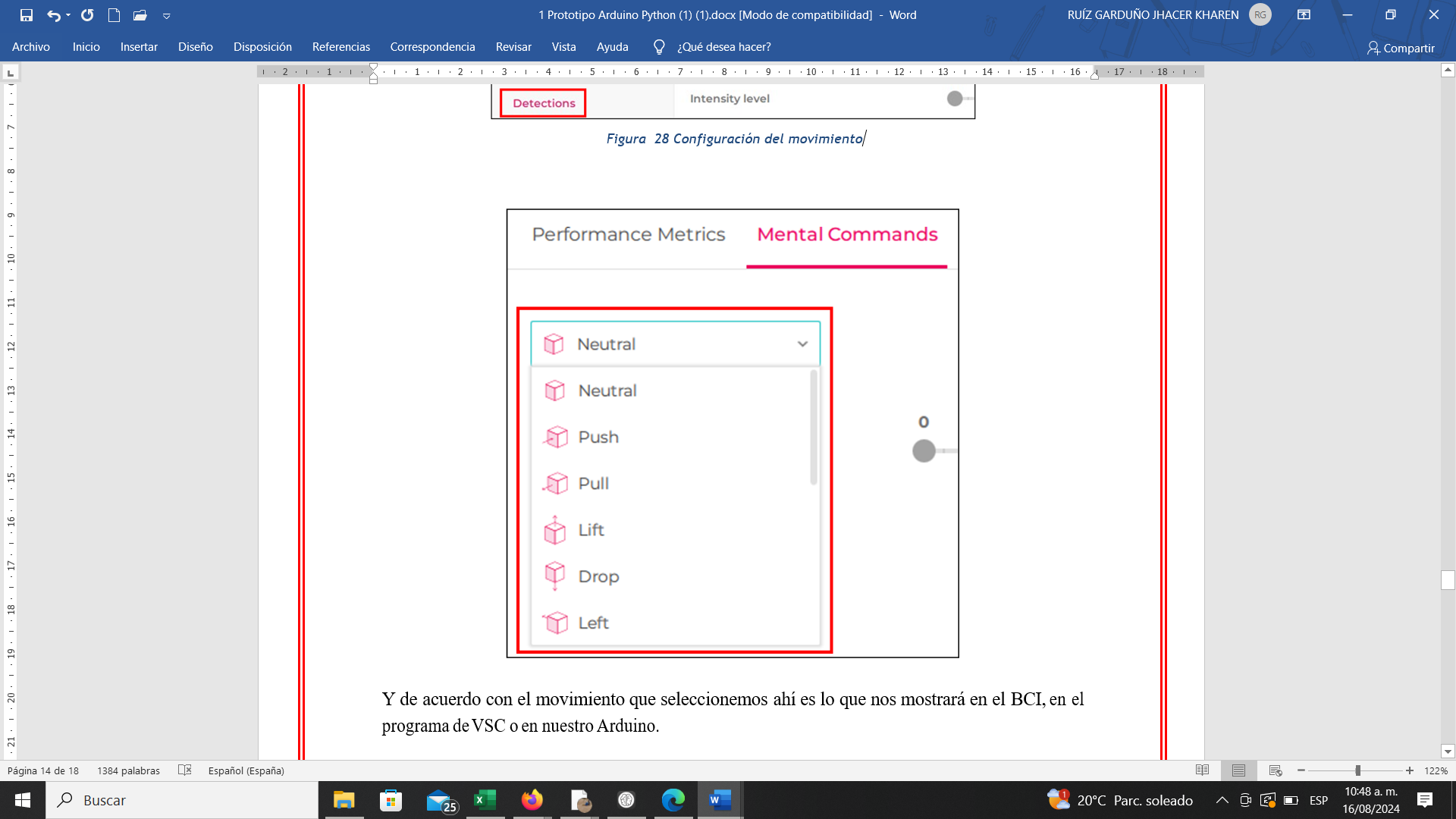


Figura 29 Tipos de movimiento

Y de acuerdo con el movimiento que seleccionemos de acuerdo al elegido mostrará en el BCI, en el programa de VSC o en nuestro Arduino.

# Manual de instrucciones para Arduino (Carrito)

**Esquema**

## 

Figura 30 Esquema del diseño

## Texto Descripción generada automáticamenteCódigo Arduino

Figura 31 Código de configuración del Arduino (parte 1)

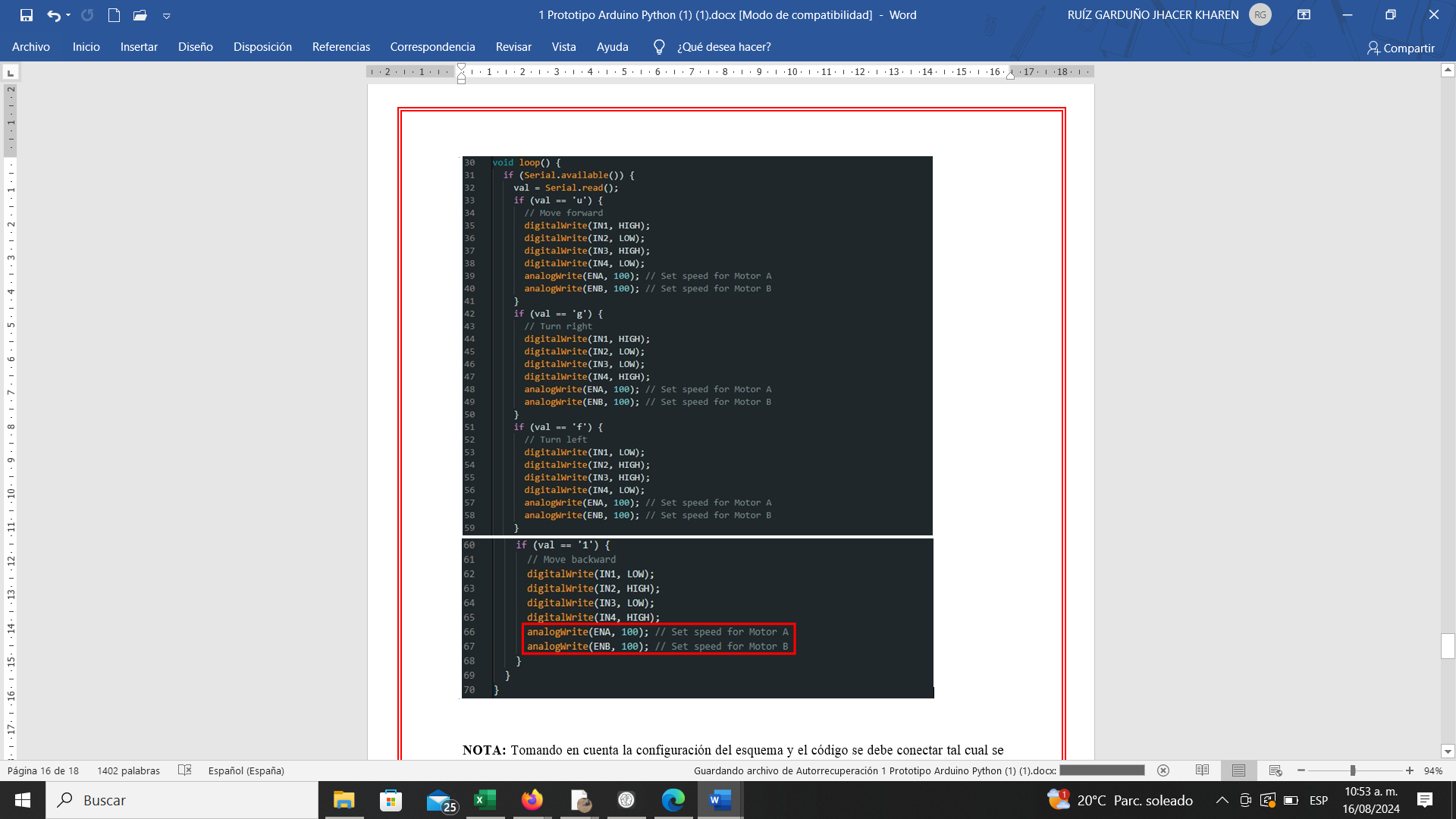


Figura 32 Código de configuración del Arduino (parte 2)

**NOTA:** Tomando en cuenta la configuración del esquema y el código se debe conectar tal cual se muestra en la imagen 32, esto puede ser en un Arduino UNO o NANO, siempre verificando que se usan los **pines digitales.**

Dentro del código Arduino, las variables **ENA** y **ENB** (marcadas en cuadro rojo), se pueden modificar sus valores desde un 1 hasta 255 dependiendo la velocidad que requiera para los motores (entre más alto el valor, más rápido será su movimiento).

En el ejemplo del código se manejan valores de 100, una velocidad considerable para probar nuestro carrito.

## Código para modificar en Python

Dentro del código en Python se realizan las siguientes modificaciones.

**Paso 1:** En la línea **221** será el método en donde agregará lo siguiente:

## 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

## 

Figura 33 Definición de variables

La parte encerrada en amarillo es el bloque que debe identificar y el recuadro rojo es el código que agrega, el cual se encargará de leer los datos de la diadema y mandar el movimiento al Arduino.

**NOTA:** A la hora de copiar el código y ponerlo en el método es muy importante respetar las sangrías de cada función como se muestra en la imagen, de lo contrarió nos marcará error en el código.

De igual manera, esto se puede probar con la diadema virtual para hacer las respectivas pruebas antes de poder usar la diadema física.

**Posibles errores**

Al momento de empezar a instalar las librerías puede generar el siguiente error y es porque se debe actualizar el *pip*, y en pantalla mostrará el comando que se debe utilizar para hacerlo.

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

Figura 34 Error de pantalla