UNIVERSIDAD DE EL SALVADOR

FACULTAD DE INGENIERÍA Y ARQUITECTURA

ESCUELA DE INGENIERÍA ELÉCTRICA

CONTROL REMOTO DE UNA MESA PARA ANALIZAR RESONANCIA EN ESTRUCTURAS

PRESENTADO POR:

SALGUERO GARCÍA, CÉSAR IVÁN

PARA OPTAR AL TÍTULO DE:

INGENIERO ELECTRICISTA

CIUDAD UNIVERSITARIA, MAYO 2022

UNIVERSIDAD DE EL SALVADOR

RECTOR:

MSC. ROGER ARMANDO ARIAS ALVARADO

FACULTAD DE INGENIERÍA Y ARQUITECTURA

DECANO:

PhD. EDGAR ARMANDO PEÑA FIGUEROA

SECRETARIO:

ING. JULIO ALBERTO PORTILLO

ESCUELA DE INGENIERÍA ELÉCTRICA

DIRECTOR:

ING. ARMANDO MARTÍNEZ CALDERÓN

UNIVERSIDAD DE EL SALVADOR

FACULTAD DE INGENIERÍA Y ARQUITECTURA

ESCUELA DE INGENIERÍA ELÉCTRICA

TRABAJO DE GRADUACION PREVIO A LA OPCION AL GRADO DE

INGENIERO ELECTRICISTA

Titulo:

CONTROL REMOTO DE UNA MESA PARA ANALIZAR RESONANCIA EN ESTRUCTURAS

PRESENTADO POR:

SALGUERO GARCÍA, CÉSAR IVÁN

TRABAJO DE GRADUACIÓN APROBADO POR:

DOCENTE ASESOR:

Dr. CARLOS EUGENIO MARTINEZ CRUZ

CIUDAD UNIVERSITARIA, MAYO 2022

TRABAJO DE GRADUACIÓN APROBADO POR:

DOCENTE ASESOR:

Dr. CARLOS EUGENIO MARTINEZ CRUZ

NOTA Y DEFENSA FINAL

En esta fecha, viernes 1 de abril de 2022, en la Sala de Lectura de la Escuela de Ingeniería Eléctrica, a las 5:00 p.m. horas, en presencia de las siguientes autoridades de la Escuela de Ingeniería Eléctrica de la Universidad de El Salvador:

1. Ing. Armando Martínez Calderón
   Director

   Firma

2. MSc. José Wilber Calderón Urrutia
   Secretario

   Firma

Y, con el Honorable Jurado de Evaluación integrado por las personas siguientes:

- DR. CARLOS EUGENIO MARTINEZ CRUZ
  (Docente Asesor)

  Firma

- MSC. JOSE WILBER CALDERON URRUTIA

  Firma

- MSC. SALVADOR DE JESUS GERMAN

  Firma

Se efectuó la defensa final reglamentaria del Trabajo de Graduación:

CONTROL REMOTO DE UNA MESA PARA ANALIZAR RESONANCIA EN ESTRUCTURAS

A cargo del Bachiller:

- SALGUERO GARCÍA CÉSAR IVÁN

Habiendo obtenido en el presente Trabajo una nota promedio de la defensa final: 8.4

(Ocho punto cuatro)

AGRADECIMIENTOS.

Agradezco en primer lugar a Dios, por haberme brindado la sabiduría y fortaleza para culminar este trabajo de graduación y haber logrado completar mi carrera universitaria. Él ha sido parte fundamental para mantener la calma y motivación ante todas las dificultades que hubo durante este camino.

En segundo lugar, agradezco y dedico este título a mi familia, la cual siempre me apoyo y me brindo la oportunidad de poder estudiar una carrera universitaria. Gracias a mi padre César Salguero por todo el sacrificio realizado durante estos años, por siempre trabajar y darnos la mejor vida que ha sido capaz de ofrecernos a mí y mi familia. Gracias a mi madre Santos Marta García por ser la que me animaba y recordaba de estudiar siempre que no tenía la motivación necesaria para hacerlo.

Agradezco a mi asesor de tesis el ingeniero Carlos Martínez por siempre estar pendiente y motivarme a continuar con el trabajo de graduación, sin el este camino hubiera sido mucho más difícil de afrontar.

Gracias a mis compañeros de estudio Carlos Campos, Luis Parada, Marco Diaz, Oliver Padilla, Daniel Zelaya, Andrés García, y demás personas que me formaron parte de mi proceso de aprendizaje, los cuales me acompañaron durante los últimos años de la carrera y se convirtieron en algunos de mis más grandes amigos, los cuales espero que me acompañen durante todo el crecimiento de mi vida profesional.

Gracias a mi novia Gabriela Villagrán por ser mi principal fuente de motivación durante los últimos años de carrera, gracias a ella he podido esforzarme más y realizar el mejor trabajo posible.

Agradezco en general a mi familia, hermanos, primos, tíos, etc. Por siempre estar pendientes de mi avance por la universidad, por ayudarme en todas las ocasiones donde les fue posible y lo necesite, y a grandes rasgos por el acompañamiento que me dieron siempre.

Agradezco al ing. Pocasangre y Sra. Reina Vides por la asesoría para el desarrollo de la tesis y los trámites necesarios para culminar con este trabajo de graduación. Y en general agradezco a todos mis profesores por formar parte de mi proceso de educación para la obtención de este título de Ingeniero Electricista.

CONTENIDO

ILUSTRACIONES

RESUMEN EJECUTIVO

El presente trabajo de graduación describe los aspectos técnicos necesarios utilizados para modificar y adaptar una mesa móvil simuladora de resonancia desarrollada en la Escuela de Ingeniería Eléctrica de la Universidad de El Salvador. El control de la mesa se realiza a través de cuadernos de trabajo escritos en lenguaje Python utilizando un entorno informático interactivo web llamado *Jupyter Notebook*. Este desarrollo basado en la Internet permite el control de múltiples usuarios sin necesidad de tener una conexión física directa. De esta manera se consigue utilizar el simulador de forma más eficiente, pudiendo realizar prácticas de laboratorio de manera simultánea, reduciendo el número de mesas móviles a utilizar.

Además, se especifican los aspectos técnicos relacionados a la construcción de la mesa móvil, es decir los materiales, mecanismo de movimiento, hardware, etc. De igual manera se incluye la guía de instalación de todos los programas necesarios para establecer el servicio de jupyter notebook de forma remota, es decir las líneas de comando utilizadas, las configuraciones adicionales. Y finalmente se describen las rutinas en lenguaje Python utilizadas para el control de la mesa, en donde se explican las generalidades del código y las partes claves de este.

CAPÍTULO 1

1.1 INTRODUCCIÓN

Con el objetivo de extender y facilitar el estudio acerca del análisis de señales sísmicas en la Escuela de Ingeniería Eléctrica de la Universidad de El Salvador se construyó un prototipo de mesa de pruebas, cuya función era generar señales analógicas controladas que puedan medirse mediante un acelerómetro. Está "mesa móvil" como la llamaremos de aquí en adelante, era controlada mediante un módulo ESP32 y el puerto serial desde un único usuario por equipo, esto inicialmente limitaba el alcance que podía tener un único equipo para el uso en laboratorio ya que idealmente se necesitaría un equipo de mesa móvil por grupo de trabajo.

Debido a esto a lo largo del documento se desarrolla un método con el cual se podrá utilizar el equipo de manera remota por otros usuarios, siempre necesitara estar conectado mediante el puerto serial a una computadora, pero a la vez podrá ser utilizada mediante internet con la plataforma llamada *Jupyter Notebook* sin estar conectado físicamente a la mesa móvil, lo cual permitirá que pueda ser utilizada simultáneamente por varios equipos de trabajo en un laboratorio.

El análisis de señales posee múltiples aplicaciones en la vida real, ya que nos permite estudiar un sinfín de fenómenos que son de vital importancia no solo en el área didáctica, sino en el área industrial, telecomunicación, construcción, etc. Algunos ejemplos de estos son la diagnosis de rodamiento, la cual comúnmente se utiliza en el área industrial para el mantenimiento preventivo de motores, en el cual se miden las vibraciones de estos, o en el área automotriz donde se miden las vibraciones de los amortiguadores o del chasis y sirve para evitar accidentes al facilitar la identificación de piezas o partes en mal estado, en el área de medicinas se utiliza en equipos médicos como resonancia magnética, tomografías, ecogramas, etc. Que son equipos en los cuales es necesario procesar imágenes generadas por ellos para tener una mejor visión de las mismas. Actualmente en la Escuela de ingeniería Eléctrica solo una asignatura se dedica específicamente al estudio de esta rama, por lo que este trabajo de graduación presenta un esfuerzo para la ampliación de estos conocimientos y a la vez facilitar un poco su estudio.

1.2 INTERÉS DE LA INVESTIGACIÓN

El desarrollo de esta investigación es motivado por el interés en desarrollar un sistema de control remoto que permita facilitar el uso de la mesa móvil de resonancia natural. De esta manera se podrían seguir realizando prácticas de laboratorio sin la necesidad de que un estudiante esté presente en el laboratorio, o que varias personas puedan trabajar en simultáneo con un mismo equipo.

Ya que este es un equipo pensado para trabajar en el laboratorio con el cual se pueda estudiar el análisis de señales sísmicas por lo estudiantes se considera importante que pueda ser fácilmente replicada y cuya inversión se mantenga lo más baja posible. La mesa móvil facilitara el entendimiento del fenómeno de resonancia en estructura y su análisis de señal.

1.3 ANTECEDENTES

El primer antecedente encontrado dentro de la Escuela de Ingeniería Eléctrica de la Universidad de El Salvador de este trabajo de graduación parte de una maqueta básica que se empezó a desarrollar dentro de la asignatura “Aplicaciones del Tratamiento Digital de la Señal”, en ese momento se construyó una maqueta para hacer oscilar estructuras. Durante esa etapa el objetivo fue la visualización de una estructura sometida a señales de diferentes frecuencias, aunque el objetivo se logró no era posible medir ese efecto ni tener mayor control sobre la señal generada.

Posteriormente la investigación se continuó en “Proyecto de Ingeniería Eléctrica” [1], donde se construyó un primer prototipo de mesa de pruebas para el estudio de resonancia en estructuras.

Durante esta etapa se probaron distintas combinaciones, materiales y formas en las cuales podría ser construida mecánicamente la mesa de pruebas, todo para garantizar que el movimiento fuera lo más cercano posible a una onda senoidal. Inicialmente se propuso un mecanismo de manivela para generar el movimiento tal y como aparece en la ilustración 1, sin embargo, como nos daríamos cuenta posteriormente este resultó ser un método poco efectivo para producir el movimiento.

Además de esto se analizó el tipo de motor que se utilizaría, el cual hasta este momento es un motor DC que fue escogido por facilidad ya que se poseía con anterioridad.

También se diseñó la etapa de control la cual consiste en un Arduino y unos sensores los cuales permitían un control preciso del giro del motor para controlar la frecuencia y las oscilaciones de la mesa. La etapa de medición y tratamiento de la señal se realizó extrayendo los datos del acelerómetro manualmente del Arduino, para luego ser procesados mediante Matlab. Aquello resultaba lento y poco intuitivo para poder estudiar el fenómeno de resonancia.

Ilustración 1: Mesa móvil durante proyecto de ingeniería 1.

Esta misma investigación se continuó durante la asignatura "Proyecto de Ingeniería Eléctrica 2", en la Ilustración 2 podemos observar la construcción de un nuevo prototipo de mesa de pruebas [2]. Durante la elaboración de este nuevo proyecto se diseñaron estructuras que permitieran observar fácilmente el efecto de la resonancia, es decir que oscilan fácilmente a la frecuencia correcta, además se modificó el programa para que ahora los datos del acelerómetro se transmiten mediante el puerto serial a la computadora y esta pueda realizar el tratamiento y análisis de la señal en tiempo real, esto permitió un mejor y más rápido estudio del fenómeno. Con esos cambios se mejoró la visualización pudiendo verse lo que sucedía en la estructura gráficamente. Al realizar estas mejoras en el programa y la utilización de la mesa se tuvo que utilizar otro módulo Arduino debido a que uno solo no podía con todas las rutinas de interrupción que se utilizaban. También, se cambió el mecanismo de movimiento de la mesa, en esta ocasión se decidió utilizar un tornillo infinito y rieles, los cuales demostraron ser más efectivos y proporcionaron un movimiento más suave y limpio que redujeron en gran medida los errores de medición.

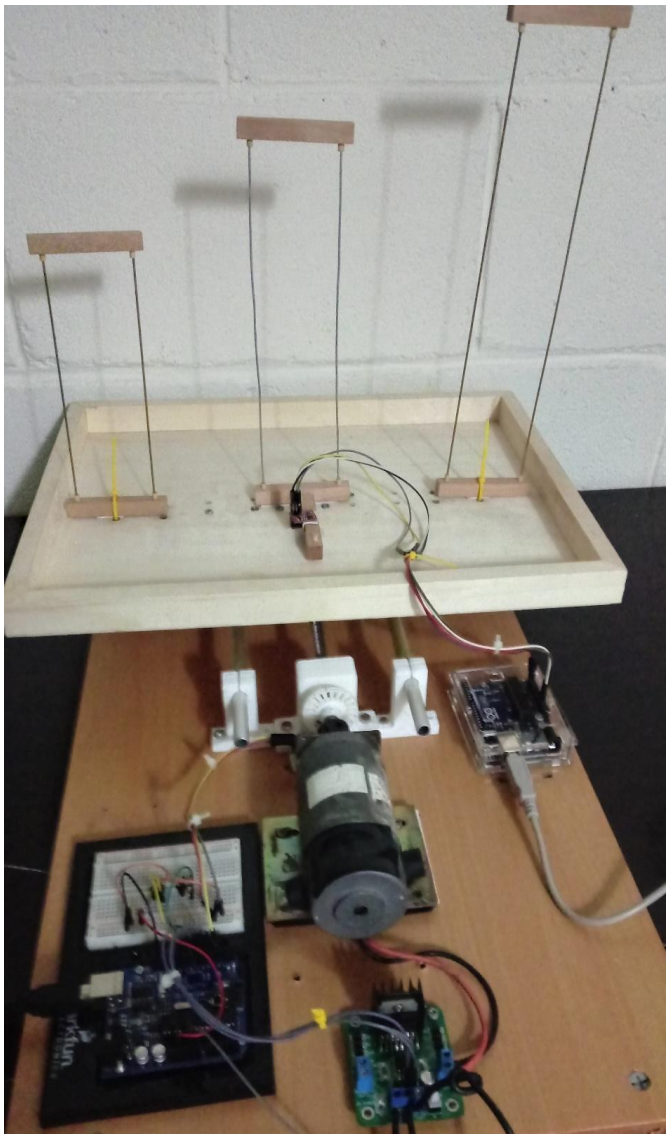

Ilustración 2: Mesa móvil durante proyecto de ingeniera 2

Y finalmente se continuo su desarrollo a lo largo de un trabajo de graduación [3] en donde se mejoró la etapa de control al utilizar un ESP32 para el control de la mesa en lugar de 2 módulos Arduino. Mejorando significativamente el uso de la mesa ya que ahora en lugar de tener que programar 2 módulos Arduino diferentes y utilizar 2 programas distintos ya se puede trabajar solamente con uno. Facilitando así el entendimiento de los programas y de cómo funciona la mesa, de esta forma permite modificar los códigos más fácilmente y de cierta manera el equipo funciona más eficientemente, ya que ahora no es necesario sincronizar ambos microcontroladores. Esto es posible gracias a que el ESP32 posee 2 núcleos independientes, por lo que permite la creación de rutinas más complejas y disminuye el error debido a que ahora cada proceso puede ser desarrollado por un núcleo

diferente. Además, hubo mejoras significativas en la parte de la medición al utilizar un acelerómetro y un geófono. En donde quedo demostrado que las mediciones realizadas mediante el geófono fueron más exactas que las del acelerómetro. Respecto al apartado de análisis de señales se pudo realizar la transformada de Fourier en tiempo real, lo que facilitaba la comprensión del fenómeno y hacia el análisis de resultados bastante atractivo a la vista.

Se mejoró el movimiento al implementar un tornillo infinito y sus soportes tipo eje para el movimiento, esto ya que anteriormente se utilizaban unos soportes y adaptadores creados en una impresora 3D lo que provocaba vibraciones en el movimiento. Para el control del motor se utilizó un *encoder* con el cual se configuraba la velocidad y ángulo de movimiento del motor, de esta manera se le daba un mayor grado de exactitud a la configuración del movimiento en el programa en el ESP32.

La ilustración 3 muestra el diseño de la mesa móvil con todas estas mejoras mencionadas anteriormente. Para la configuración de la frecuencia a la cual trabajaba la mesa móvil se utilizó una botonera, la cual presentaba varias limitantes, el primero es que se establecía un rango en el cual iba variando la frecuencia, por lo que la precisión con la que podía ser configurada era más limitada. A esto se le suman los problemas por el mismo hardware, en donde no siempre se leía la señal proveniente de la botonera y era necesario el oprimir el botón múltiples veces para lograr que funcionara.

Otra limitante del equipo es que no se podían guardar las mediciones generadas, por lo que para realizar el tratamiento de la información solo podía hacerse durante el funcionamiento de la mesa móvil. De igual manera ya que el ESP32 tenía que ser programado y luego ser llamado para poder empezar a funcionar era poco intuitivo para poder realizar modificaciones en el código. Además, era necesario tener instalado un programador de Arduino para modificar o correr el código del programa.

Muchas de estas limitaciones mencionadas son las que ahora durante el desarrollo de este trabajo de graduación se tratan de solucionar o mejorar.

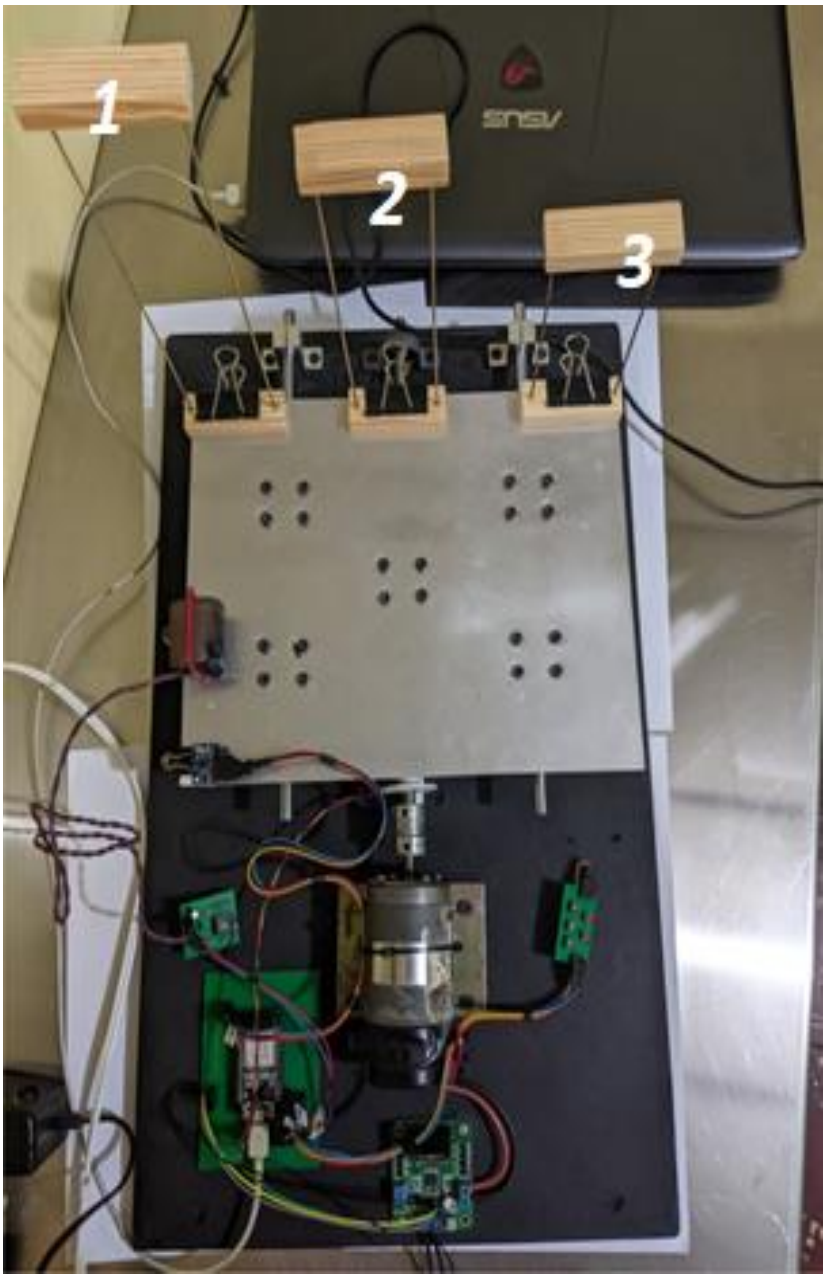

Ilustración 3: Mesa móvil durante el trabajo de graduación "Diseño y construcción de un simulador de resonancia en estructuras" [3]

1.4 OBJETIVOS

1.4.1 OBJETIVO GENERAL

Controlar de forma remota una mesa de pruebas de aceleración de estructuras.

1.4.2 OBJETIVOS ESPECÍFICOS

- Modificar los parámetros de control de un motor DC de manera remota.

- Crear un servidor, página web o documentos web de Jupyter donde se pueda acceder fácilmente desde cualquier dispositivo con conexión a internet.

- Desarrollar y ampliar la algoritmia necesaria para el tratamiento de la resonancia en estructuras debido a las aceleraciones.

1.5 ORGANIZACIÓN

Inicialmente se encontraron varios puntos de mejora en la mesa prototipo, sin embargo, la parte medular de este trabajo de graduación es lograr controlar el dispositivo remotamente. Por ello, se presentan tres grandes apartados para la implementación del sistema de control remoto a la mesa simuladora de resonancia natural, el primero presenta los cambios previos realizados para adecuar el sistema de control del equipo a una plataforma web que nos permita programar el microcontrolador mediante internet, en este caso ya que se está utilizando el microcontrolador ESP32 utilizaremos M*icropython* y su compatibilidad con *Jupyter Notebook.*

El segundo apartado corresponde a la elaboración de rutinas de interrogación y de control que permitan a la mesa operar de la misma forma que lo hacía inicialmente. Como último apartado se presenta la creación de un servidor mediante *Google Cloud*, el cual nos permitirá crear una *VPN* y conectarse a ella mediante *OpenVPN*, de esa manera será posible comunicarse y controlar el equipo de manera remota, desde cualquier conexión a internet previamente añadida al servidor.

Finalmente, se mostrarán los resultados y los posibles caminos de mejora para la continuación de este proyecto. Para explicar todo esto se divide el documento en 6 capítulos.

El capítulo 2 incluye el marco teórico, es decir toda la bibliografía necesaria y en general todos los programas, protocolos, conceptos y consideraciones que se tomaron en cuenta al momento de diseñar el nuevo sistema de la mesa móvil, además incluye las razones por las cuales se escogieron esos programas o métodos, y una breve descripción de ellos y como se aplicaron en el desarrollo del trabajo de graduación.

En el capítulo 3 se describen los detalles de construcción y modificaciones realizadas a la mesa móvil con el fin de hacerla funcional con el nuevo sistema propuesto. Se detallarán las líneas de comando utilizadas, además de los detalles adicionales a cada uno de los pasos para cambiar el firmware del microcontrolador, como por ejemplo oprimir alguno de los interruptores del microcontrolador, o los programas que son necesario instalar antes de iniciar el proceso.

Una vez preparado el microcontrolador se procede con la instalación de J*upyter Notebook*, y las modificaciones necesarias a la configuración del mismo para permitir acceso multi usuario y que este sea compatible con M*icropython. A*demás, se presentan algunas opciones de seguridad que permitirán al servicio ser más seguro y tener mayor control sobre él.

Y finalmente se explican cómo utilizar los programas y servicios de *Openvpn* y *Google Cloud* para la creación de una red privada que permita utilizar el servicio de *Jupyter Notebook* en distintas redes.

En el capítulo 4 se presentan los resultados obtenidos a partir de las modificaciones realizadas, y se muestra su funcionamiento, es decir que se explican y describen cada una de las partes de los programas diseñados para el control de la mesa móvil. De esta manera será más fácil el poder modificar estos *notebooks* para diseñar distintas aplicaciones y/o pruebas que puedan ser utilizados en el laboratorio a futuro.

Además, se presenta la manera en la que actualmente se interpreta la información de los datos obtenidos a partir de las mediciones, y como utilizamos *Jupyter Notebook* para esto. De igual manera que para los notebooks de control del equipo también se explican las diferentes partes del código y el porqué de analizar la información de esta manera.

Y finalmente el capítulo 5 explica la línea futura del proyecto y como este puede seguirse modificando y mejorando. Ya que la mesa móvil tiene pensado ser utilizado en prácticas de laboratorio dentro de la Universidad de El Salvador es necesario seguir profundizando en el estudio de estas señales, y como pueden surgir varias prácticas de laboratorio a partir de ella. Y ya que este es un proyecto multidisciplinario hay muchas áreas en las cuales se puede seguir desarrollando este tema, por lo que este capítulo presenta y explica algunas de esas propuestas.

CAPÍTULO 2

A lo largo de este capítulo se explican todos aquellos conceptos necesarios que se aplicaron para las modificaciones realizadas a la mesa móvil, además en la mayoría de ellos se da una breve explicación del porqué se utilizó ese concepto y en qué momento se aplicó.

2.1 DESCRIPCIÓN GENERAL

La mesa prototipo cuenta con un sistema de control basado en un ESP32 el cual es programado mediante la *IDE* de Arduino, sin embargo, es necesario auxiliarnos de otro *IDE* que permita hacer *multihosting* para poder programarlo desde varios puntos de accesos al mismo tiempo. Además, actualmente se ha programado mediante lenguaje C++, sin embargo, el ESP32 permite ser programado mediante *Python*, por lo que al seleccionar otro IDE podremos utilizar ambos lenguajes de programación.

2.2 JUPYTER NOTEBOOK

Jupyter Notebook es una aplicación de código abierto cliente-servidor que permite el desarrollo de software en diferentes lenguajes de programación [4], esta herramienta resulta muy útil para la aplicación que estamos desarrollando ya que como se mencionara a continuación presenta distintas ventajas que facilitaran el funcionamiento de la mesa prototipo y por las cuales se escogió para utilizarlo en este trabajo de graduación.

1. Se pueden guardar y correr segmentos de código del programa individualmente, de esta manera se pueden crear rutinas de interrogación y editar directamente el código que estará corriendo en el microcontrolador.
2. Compatibilidad con el ESP32, de esta manera se puede programar el microcontrolador directa e instantáneamente desde el *notebook*.
3. Permite montar entornos multiusuarios, por lo que se puede acceder a él mediante cualquier navegador que posea la dirección IP y el puerto por el cual se brinda el servicio.
4. Permite la creación de una biblioteca en línea, por lo que cualquier programa, o avance puede ser fácilmente cargado con solo acceder al servicio.
5. Exceptuando el equipo que pondrá en marcha el servicio no es necesario la instalación de *jupyter Notebook* en ningún otro dispositivo para utilizarlo.
6. Permite la configuración de usuario y contraseña, además de otras medidas para garantizar la seguridad del servicio que se quiere proveer.

2.3 MICRO PYTHON

Para poder programar el microcontrolador mediante J*upyter Notebook* es necesario hacerlo mediante la extensión de M*icropython*, esto en necesario hacerlo ya que es la única manera en que se podrá programar el microcontrolador desde la plataforma de *Jupyter Notebook.*

Es una adecuación de P*ython 3* [5] para poder ser utilizada en microcontroladores. En palabras sencillas es un compilador que transforma el lenguaje *Python* a B*ytecode*. *Jupyter Notebook* es compatible con este módulo y permite programar el microcontrolador mediante el puerto serial, de esta forma se puede programar y ejecutar el programa de forma inmediata en el microcontrolador.

Para que el microcontrolador funcione con este lenguaje de programación es necesario instalarle el firmware de M*icropython* mediante *esptool.py*, además habrá que instalar el Kernel de M*icropython* para *Jupyter Notebook*.

2.4 UPYCRAFT IDE

Este programa es otra opción para poder escribir, compilar y programar un *ESP 32* [6] con *Micropython* de forma local, específicamente se utilizó este programa para instalar y modificar librerías dentro del microcontrolador, ya que J*upyter Notebook* al no ser una plataforma totalmente dedicada a utilizar *micro Python* no es compatible con algunos de los comandos o librerías de P*ython*. Por lo que se utilizó este otro software para instalar la librería del acelerómetro *mpu6050* y además poder editarla. De igual manera *Upycraft* permite identificar los errores y enumera las líneas de código, por lo que resulta más fácil escribir y depurar los códigos de *Micropython* en esta plataforma y luego controlar el microcontrolador mediante *Jupyter Notebook*.

2.5 PUERTOS Y SERVICIOS WEB

Jupyter Notebook funciona al proveer un servicio web a través de uno de los puertos del router, por lo que para acceder a este servicio es necesario que se cumplan las siguientes condiciones:

1. Tener acceso a un router.
2. El router debe permitir poder transmitir por un puerto.
3. El puerto no debe estar siendo utilizado por otro servicio.

2.6 CONTROL REMOTO

Hoy en día tener control remoto sobre los equipos se ha vuelto una característica casi obligatoria para tener control sobre algún proceso, este método de control utiliza la conexión de diferentes estaciones remotas asociadas a uno o varios centros de control, para esto es necesario la creación de redes de comunicación, ya sea públicas o privadas.

Para este caso en específico nos apoyamos de la red de Google para la creación de un servidor y luego establecer una *virtual private network*, se hizo de esta manera ya que no tenemos acceso directo a la red pública de ningún proveedor de internet. Además, Google posee ciertos servicios que resultan realmente útiles para el almacenamiento de información (como Google Drive), y del cual nos apoyaremos para guardar las mediciones obtenidas del acelerómetro en cada simulación.

Un aspecto importante al momento de implementar un sistema de control remoto a un equipo ya existente es intentar realizar la menor cantidad de cambios posibles al hardware, esto con el propósito de mantener la inversión lo más baja posible y no cambiar por completo el equipo.

2.7 GOOGLE CLOUD

Google cloud es una plataforma que reúne la mayoría de aplicaciones de desarrollo web de Google, prácticamente utilizaremos esta plataforma como un espacio virtual mediante el cual podremos crear un servidor y almacenar información en él [7]. De esta manera podremos interconectar diferentes redes sin necesidad de hardware especial. Este servicio funciona al crear una instancia virtual en donde se escogió un *virtual machine* de Ubuntu ya que es la opción gratuita que permite *Google cloud*. A este servicio se le llama *Compute Engine* y prácticamente es un servicio que te permite ser host, almacenar y procesar información. A la instancia virtual se le asigna una ip pública y privada, y mediante el programa de *OPEN VPN* podemos crear una *vpn* entre el servidor y los usuarios utilizando esta IP.

2.8 OPEN VPN

Este programa permite habilitar una red *VPN* para Windows, de esta manera nos podemos conectar al servidor creado en *Google cloud* solamente instalando *OPEN VPN CONNECT* en la máquina virtual, en donde se crearán los usuarios que estarán habilitados para conectarse a la *VPN* y *OPEN VPN GUI* en los equipos de los usuarios, para conectarse es necesario tener el archivo habilitador que crea la máquina virtual por lo que la primera vez que se establece la conexión es necesario que la persona tenga acceso a *Google cloud*, o en su defecto que el administrador del servidor comparta los archivos.

Otra ventaja de esta aplicación es que está disponible para Android, por lo que será posible controlar la mesa simuladora incluso desde el celular o una tableta [8].

2.9 GOOGLE DRIVE

Google drive es un servicio de almacenamiento de datos que proporciona Google, al igual que la mayoría de servicio tiene una capacidad gratis limitada de 15gb, sin embargo, esto resulta ser suficiente para nuestros fines ya que inicialmente solo se busca guardar los resultados de las mediciones del acelerómetro, es decir que solo serán algunos datos numéricos. La manera en que estos datos serán enviados del acelerómetro a *Google drive* es a través de otro servicio en línea llamado *IFTTT*, este permite recibir realizar una acción programada al recibir una solicitud web, no podemos enviar directamente los datos a *Google drive* debido a que *micro Python* no reconoce todas las librerías de *Python*, por lo que nos tendremos que auxiliar de otro servicio externo para realizar esto.

2.10 IFTTT

Es un tipo de servicio web que permite crear y programar acciones para automatizar diferentes tareas vía Interne. Al crear una *Applet* podemos indicarle la cantidad de datos que recibirá, y que al recibirlos automáticamente genere un documento en *Google drive* y los guarde ahí, de esta manera en J*upyter Notebook* se pueden enviar estos datos mediante una *request*, ya que esta función viene incluida en la librería *network* que sí está adaptada a M*icropython*.

2.11 RESONANCIA NATURAL

Ya que el objeto de estudio de la mesa móvil es la resonancia en estructura es necesario tener claro el fenómeno que se está analizando, la resonancia es un fenómeno que amplifica una vibración. Se produce cuando una vibración se transmite a otro objeto cuya frecuencia natural es igual o muy cercana a la de la fuente, en ese momento es cuando la amplitud alcanza su punto máximo en un sistema elástico.

Para definir este concepto matemáticamente primero es necesario tener claro los siguientes conceptos:

1. Frecuencia: Es la cuantificación del número de veces en que se repite un evento por unidad de tiempo, en nuestro caso será la frecuencia a la que es motor produzca el movimiento, y que podrá ser programada directamente mediante *Jupyter Notebook*.

2. Amplitud: Se define como la medida de variación máxima en un movimiento oscilatorio. Estas serán medidas a través del acelerómetro.

Para entender este fenómeno nos apoyaremos de la ecuación que define la dinámica de una masa acoplada a un resorte [9], la cual se define de la siguiente manera:

$$m\frac{d^2y}{dt^2} = -Ky \qquad (1)$$

$t = tiempo$
$m = masa$
$y = distancia$
$K = Constante\ del\ resorte$

Se define su posición en función de un movimiento armónico simple de la siguiente forma:

$$y(t) = Acos(w_o t) \qquad (2)$$

$t = tiempo$
$w_o = frecuencia\ angular$
$A = Amplitud$

Además, nos apoyaremos de la ecuación de frecuencia angular para estos sistemas, la cual es:

$$w_o = \sqrt{\frac{K}{m}} \tag{3}$$

$w_o = frecuencia\ angular$
$m = masa$
$K = Constante\ del\ resorte$

Esta frecuencia solamente depende de la constante K y de la masa m del sistema, por lo que se le conoce como la frecuencia natural del sistema, ya que no depende de la amplitud en teoría no necesita de ninguna fuerza para empezar a oscilar. Si aplicamos una fuerza externa periódica de la forma:

$$F = -Ky + Fcos(wt) \tag{4}$$

$K = Constante\ del\ resorte$
$y = distancia$
$F = fuerza\ externa$
$w = frecuencia\ externa$
$t = tiempo$

Al derivar la fuerza con respecto al tiempo nos queda la siguiente expresión:

$$m\frac{d^2y}{dt^2} = -Ky + Fcos(wt) \tag{5}$$

$K = Constante\ del\ resorte$
$F = fuerza\ externa$
$w = frecuencia\ externa$
$t = tiempo$

Y si utilizamos la misma función para definir la posición y la sustituimos en la ecuación anterior nos queda de la siguiente manera:

$$mAw^2cos(wt) = -KAcos(wt) + Fcos(wt) \tag{6}$$

$w = frecuencia\ externa$
$t = tiempo$
$K = Constante\ del\ resorte$
$A = Amplitud$
$F = fuerza\ externa$

Al despejar la amplitud obtenemos la siguiente ecuación:

$$A = \frac{F}{K - mw^2} \tag{7}$$

$A = Amplitud$
$F = fuerza\ externa$
$m = masa$
$w = frecuencia\ externa$
$K = Constante\ del\ resorte$

Escriba aquí la ecuación.

Si sustituimos K con la ecuación de la frecuencia natural.

$$A = \frac{F}{m[w_o^2 - w^2]} \tag{8}$$

$A = Amplitud$
$F = fuerza\ externa$
$m = masa$
$w = frecuencia\ externa$
$w_o = frecuencia\ angular$

Recordando que w_o es la frecuencia natural del sistema, y w es la frecuencia de la fuerza externa aplicada, cuando w tiende al valor de w_o la amplitud aumenta, este mismo principio se aplica para todo tipo de estructuras, por eso al someter un cuerpo a una fuerza externa que oscile muy cercano a la frecuencia natural del cuerpo este entra en estado de resonancia, llegando las amplitudes a su punto máximo.

2.12 INSTRUMENTO DE MEDICIÓN

Para la medición se utiliza un acelerómetro, el cual permite cuantificar la amplitud de la aceleración aplicada por el motor a la mesa, en nuestro caso se usará el acelerómetro MPU6050 [10], el cual permite medir aceleraciones en los tres ejes sin embargo ya que el movimiento que genera la mesa prototipo solo es en un eje no es necesario registrar las mediciones en los tres, por lo que será necesario editar la librería para evitar registrar todos los datos.

2.13 ESQUEMA DE CONEXIÓN

Para que un servicio pueda proveerse fuera de una red local, es necesario hacer visible el puerto donde se ofrece este servicio, esto se hace al configurar la *NAT (network address translation)* para dirigir las solicitudes externas hacia un puerto en específico, para realizar esto es necesario realizar ciertas configuraciones en el router donde se provee el servicio, lo primero es redirigir las solicitudes IP externas a un puerto en específico, es decir vincular un puerto a una dirección IP local, esto se realiza al abrir los puertos y habilitar que estos puedan ser vistos por IP externas (*Port forwarding*).

El segundo paso es la configuración del *NAT*, este es un protocolo que convierte en tiempo real las direcciones utilizadas en los paquetes transportados, de esta manera se pueden comunicar dos o más routers que sus IP no sean compatibles. El problema con esta solución es que los proveedores de internet en El Salvador no asignan IP públicas a todos sus usuarios por lo tanto el configurar la *NAT* para redirigir las solicitudes externas resulta imposible.

Como alternativa para establecer la conexión entre dos o más redes *LAN* se utiliza una conexión *VPN*, en donde se utiliza *Google Cloud* como puerta de enlace principal, es decir que actuara como un concentrador de *VPN*, trabajando como central de la red.

Este servicio de Google nos permite tener control de una IP pública, y mover los datos a través de ella, el diagrama de la ilustración 4 sirve como resumen de lo descrito anteriormente.

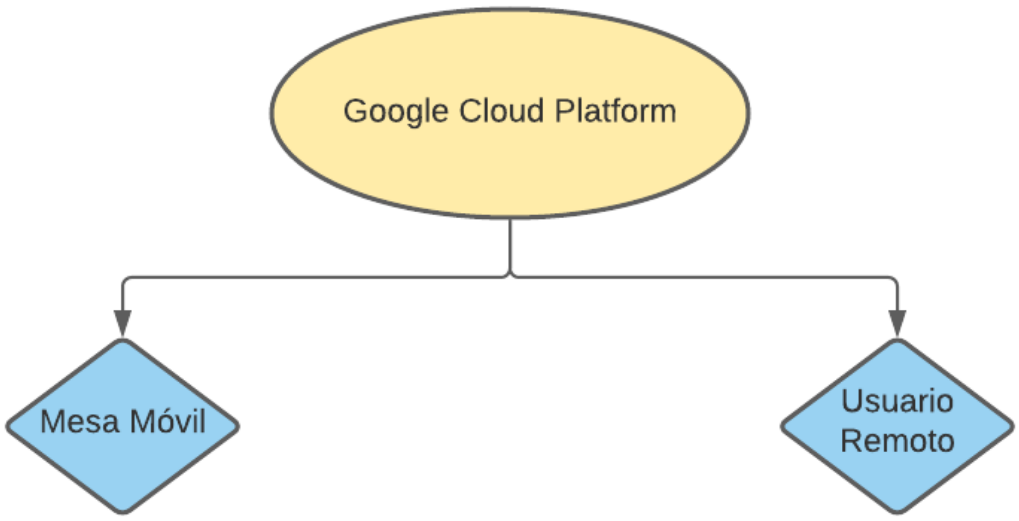

Ilustración 4: Esquema de conexión mediante Google Cloud

De esta manera logramos conectar en la misma red los usuarios y la mesa prototipo aun cuando estos están conectados a redes *LAN* diferentes inicialmente. Esta configuración permite la conexión de muchos usuarios simultáneamente, sin embargo al ser un servicio privado no es totalmente gratuito, por lo que será necesario pagar una suscripción mensual para utilizar el servidor de *google cloud*, para motivos del desarrollo de este trabajo de graduación será suficiente con el tiempo que *google cloud* se puede utilizar de manera gratuita, por lo que esta solución resulta ser muy efectiva para el cumplimiento de los objetivos del trabajo de graduación, sin embargo para la proyección del proyecto a largo plazo es preferible solicitar una ip pública o un puerto disponible en las ip públicas de la Universidad, y acceso a la configuración de sus routers para proveer el servicio.

2.14 PROTOCOLOS DE RED Y CONCEPTOS BÁSICOS.

Para entender cómo funciona la conexión mediante Google Drive y Open VPN resulta bastante útil conocer previamente los siguientes conceptos.

1. IP: Una dirección IP es una dirección única que identifica a un dispositivo en Internet o en una red local. IP significa “protocolo de Internet”, que es el conjunto de reglas que rigen el formato de los datos enviados a través de Internet o la red local. Son los identificadores que permiten el envío de información entre dispositivos en una red. Contienen información de la ubicación y brindan a los dispositivos acceso de comunicación. Internet necesita una forma de diferenciar entre distintas computadoras, enrutadores y sitios web. Las direcciones IP proporcionan una forma de hacerlo y forman una parte esencial de cómo funciona Internet. Es una cadena de números separados por puntos. Las direcciones IP se expresan como un conjunto de cuatro números, por ejemplo, 192.158.1.38. Cada número del conjunto puede variar de 0 a 255. Por lo tanto, el rango completo de direcciones IP va desde 0.0.0.0 hasta 255.255.255.255.

2. IP privada y pública: La dirección IP pública es un número único que identifica nuestra red desde el exterior, mientras que la dirección IP privada es un número único que identifica a un dispositivo conectado en nuestra red interna.

3. Puertos: Los puertos del router son las interfaces o «puertas virtuales» que utiliza un ordenador conectado a una red para la entrada y salida de datos.

4. LAN: Una LAN, abreviatura para *Local Area Network* (Red de Area Local), es una red que cubre un área geográfica pequeña, como hogares, oficinas y grupos de edificios.

5. WAN: Una WAN, abreviatura de *Wide Area Network* (Red de Area Amplia), es una red que cubre extensas áreas geográficas y que pueden abarcar todo el mundo.
6. VPN: VPN son las siglas de *Virtual Private Network*, en pocas palabras es una forma de crear una red local sin necesidad de que los usuarios estén conectados físicamente o mediante hardware especial entre sí, sino a través de internet. Una ventaja importante de este tipo de conexión es que protege el acceso de los usuarios mediante túneles de datos.
7. Túnel de datos: es la técnica que consiste en encapsular un protocolo de red sobre otro, creando un túnel de información dentro de una red de computadoras, así se puede redireccionar tráfico de datos.
8. ISP: *Internet Service Provider*, es la empresa que brinda conexión a internet a sus clientes, en el caso de El Salvador sería Claro, Tigo, etc
9. NAT: *Network Address Translation*, prácticamente es la etapa en donde las ip privadas de una LAN son convertidas a la IP pública para poder comunicarse con internet, la forma más común en donde se utiliza el NAT es a través de los puertos del router, de este modo se le vincula un puerto a cada dirección IP, de modo que cada vez que llegue información al router este redirige la información al puerto correspondiente.

2.17 PRINCIPIO DE MUESTREO DE NYQUIST

El Teorema de Muestreo de Nyquist explica la relación entre la velocidad de muestreo y la frecuencia de la señal medida [11] . Afirma que la velocidad de muestreo f_s debe ser mayor que el doble del componente de interés de frecuencia más alto en la señal medida. Esta frecuencia por lo general se conoce como la frecuencia Nyquist, f_N.

$$f_s > 2 * f_N \tag{9}$$

Si una señal es muestreada a una velocidad de muestreo menor que el doble de la frecuencia Nyquist, los componentes de frecuencias falsas más bajas aparecen en los datos muestreados. Este fenómeno se conoce como aliasing. La ilustración 5 muestra una onda sinusoidal de 800 kHz, la línea punteada indica la señal de alias registrada en una velocidad de muestreo menor a la definida por NYQUIST. La frecuencia de 800 kHz usa un alias de vuelta en el paso banda, apareciendo falsamente como una onda sinusoidal de 200 kHz.

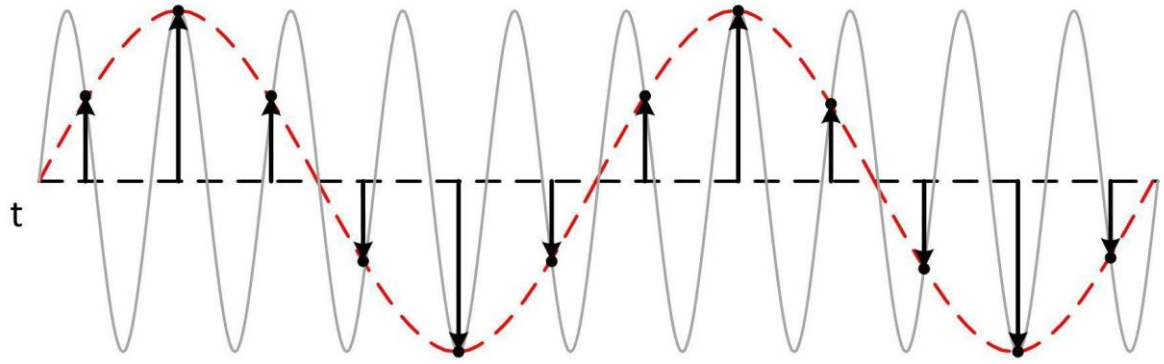

Ilustración 5: Representación de Aliasing en una onda donde el muestreo es demasiado bajo para la frecuencia de la onda, fuente: www.ni.com [12]

CAPÍTULO 3

A lo largo de este capítulo se explican la mayoría de los detalles, modificaciones y tareas que se realizaron para hacer funcional la mesa móvil. Ya que este es un equipo que involucra distintas disciplinas para su desarrollo es necesario mencionar algunas de ellas, las cuales son realmente fundamentales para el proyecto.

3.1 ESTRUCTURA MECÁNICA DE MESA MÓVIL

Una parte fundamental para que la mesa funcione correctamente es el mecanismo mecánico, ya que este es un proyecto que involucra distintas disciplinas para su desarrollo fue necesario trabajar en áreas que no competen específicamente a la ingeniería eléctrica, para que el movimiento se produjera de la mejor manera posible se probaron los siguientes mecanismos:

1- Biela manivela: Este mecanismo transforma el movimiento circular en movimiento rectilíneo alternativo, es un sistema reversible, lo que quiere decir que también puede funcionar para convertir un movimiento lineal alternativo en otro de giro. Una representación de este sistema es como el que aparece en la ilustración 6.

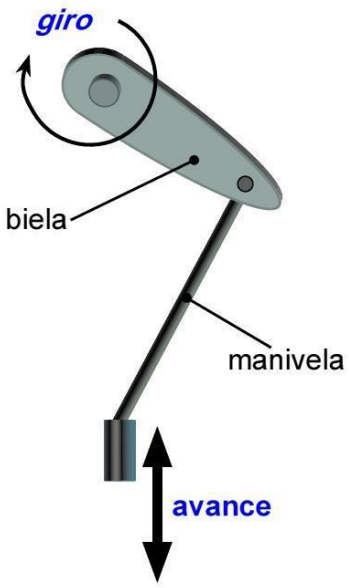

Ilustración 6: Mecanismo Biela Manivela. Imagen extraída, fuente: sitio www.edu.xunta.gal [13]

El problema con este mecanismo es que el movimiento no era aplicado directamente al centro de masa de la mesa, por lo que siempre se producía un movimiento más fuerte hacia un lado que hacia el otro.

2- Tornillo de eje óptico: Este sistema consta de 2 ejes metálicos con un diámetro de ¼ de pulgada y 20cm de longitud como el de la ilustración 7, que sirve para darle equilibrio a la superficie, y un tornillo sin fin con las mismas dimensiones con un acople que permite generar el movimiento de una manera bastante precisa. Este movimiento a diferencia del anterior debe ser en ambos sentidos, debido a que por la misma naturaleza del mecanismo este no produce un movimiento rectilíneo alternativo. Esto implica que la etapa de control del dispositivo también cambiará dependiendo del tipo de mecanismo que se utilice.

Ilustración 7: Componentes de movimiento por tornillo de eje óptico. Fuente: www.amazon.com [14]

3.2 HARDWARE DE MESA MÓVIL

Si hablamos del resto de elementos físicos que componen la mesa móvil encontraremos los siguientes:

1- ESP32: Este es el cerebro de la mesa, aquí se van interpretando las líneas de código escritas en Python mediante J*upyter Notebook*. Es un microcontrolador del tipo SoC, que significa System on a Chip, es decir que está formado por su núcleo y varios módulos en un solo circuito integrado, cuenta con tecnología *Wi-Fi* y *Bluetooth* las cuales facilitan y expanden las aplicaciones que pueden realizarse con este módulo. Existen varias características en específico por las cuales se escogió este microcontrolador, algunas de estas se mencionan a continuación:

   1. CPU: Central Processing Unit, este módulo posee 2 núcleo por lo que tiene la capacidad de realizar dos acciones simultáneamente, esto es una ventaja ya que debido a la naturaleza del movimiento que necesitamos en la cual invertimos el sentido del giro del motor y leemos y guardamos las mediciones del acelerómetro es necesario realizar estas acciones simultáneamente.

   2. Puerto PWM: Pulse Width Modulation, este puerto podría ser útil en el caso de utilizar otro tipo de motor, por lo que es perfecto para aplicaciones futuras.

   3. I^2C: Inter-Integrated Circuit, este puerto sirve para conectar periféricos y establecer comunicación entre ellos mediante el puerto seria, se utilizó para establecer comunicación con el acelerómetro.

   4. Compatibilidad con *Micropython*: Existe un firmware que permite la compatibilidad con *Micropython* con este módulo, por lo que se adapta al método utilizado para el control de la mesa.

2- Motor DC: La única característica en específico que debe tener este componente es que sea capaz de mover la mesa móvil, actualmente se utiliza un motor dc, pero esto no limita a que no puedan utilizarse otro tipo de motores que incluso podrían incluir otros beneficios adicionales como la utilización de un encoder.

3- Placa controladora de motor L298: Se posee un controlador de motores que facilita el cambio de sentido de giro del motor CD, este opera con una fuente externa y se

controla mediante pulsos provenientes del ESP32, básicamente es un puente H que facilita la etapa de programación y de construcción de un circuito extra.

4- Acelerómetro MPU6050: Este es un dispositivo que permite medir aceleraciones en los ejes X, Y, Z, en el caso de la mesa móvil ya que el movimiento se produce en una sola dirección no es necesario realizar las mediciones en los ejes Y y Z, por lo que fueron deshabilitados directamente de la librería del módulo. Para este caso se utilizó el acelerómetro

3.3 ETAPA DE CONTROL

Como se mencionó durante el capítulo 2 para la utilización de M*icropython* es necesario cambiar el *firmware* del ESP32 antes de empezar a escribir el programa en python. Para hacer esto será necesario una computadora, el microcontrolador y la herramienta *esptool*, la cual permite formatear e instalar el nuevo *firmware* en el microcontrolador. La mayoría de las veces es necesario oprimir el botón *boot* en el microcontrolador antes de poder flashear la unidad, de igual manera se necesitará instalar el *driver* correspondiente para que la computadora reconozca y puerta asignarle un puerto serial al ESP32, de lo contrario ni esptool ni la computadora reconocerán el dispositivo. Todas las rutinas, líneas de comando y detalles de la instalación y uso serán anexadas al final del documento en la sección de Anexos.

Posteriormente se realiza la instalación de *Jupyter Notebook*, esto se hace a través de la ventana de comandos. Y por último será necesario la instalación del *kernel* de M*icrophyton* para *Jupyter Notebook* [15]. Para iniciar el servicio basta con escribir “Jupyter Notebook” en la ventana de comandos tal y como se muestra en la ilustración 8.

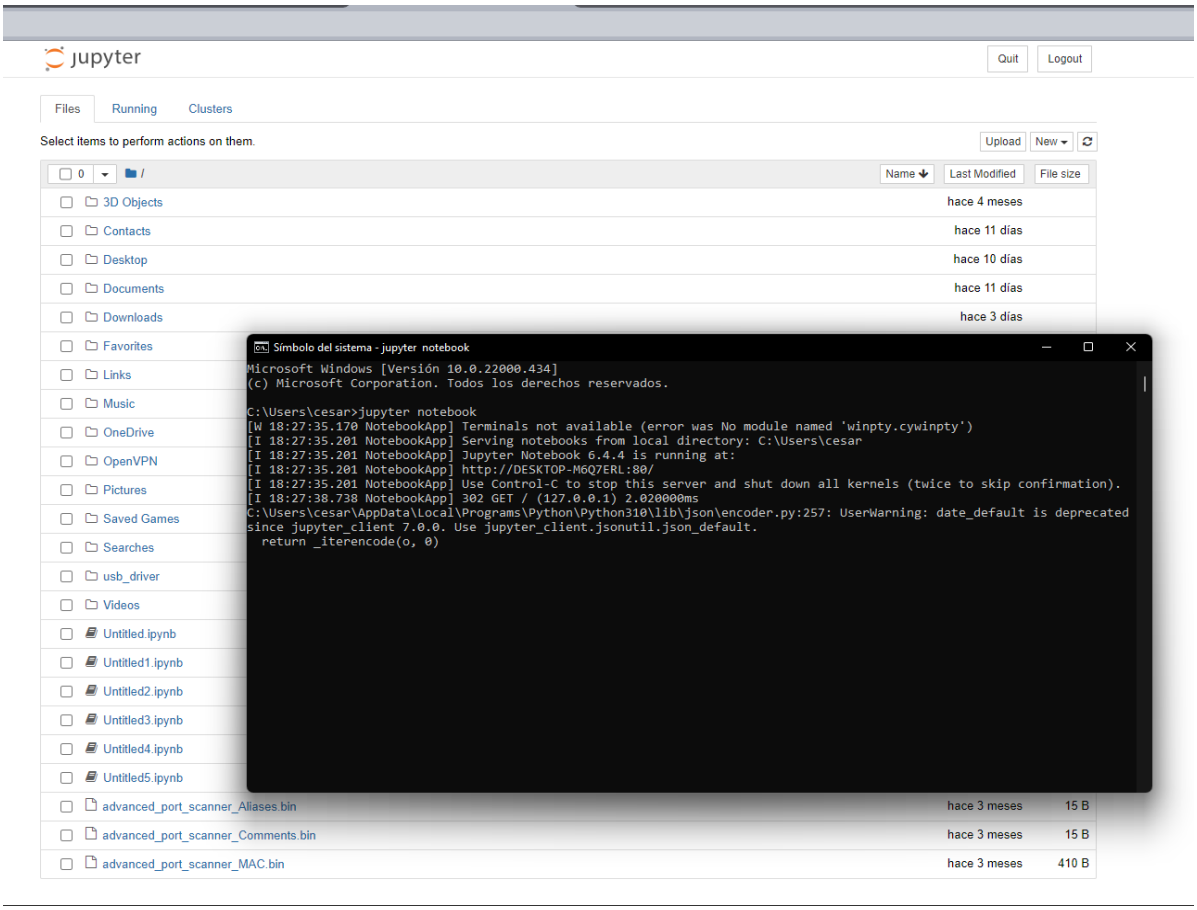

Ilustración 8: Jupyter Notebook iniciado desde ventana de comandos

Una vez hecho esto ya se puede utilizar el microcontrolador con J*upyter Notebook* mediante la computadora que está conectado directamente a él, ya que en el apartado de *new* ahora aparecerá la opción de *MicroPython USB*. Para que otros usuarios de la misma red *LAN* puedan acceder a este servicio es necesario configurarlo para que pueda recibir solicitudes de cualquier dirección IP, esto se hace al entrar en el archivo de configuración de *Jupyter Notebook* y modificarlos directamente [16] [17].

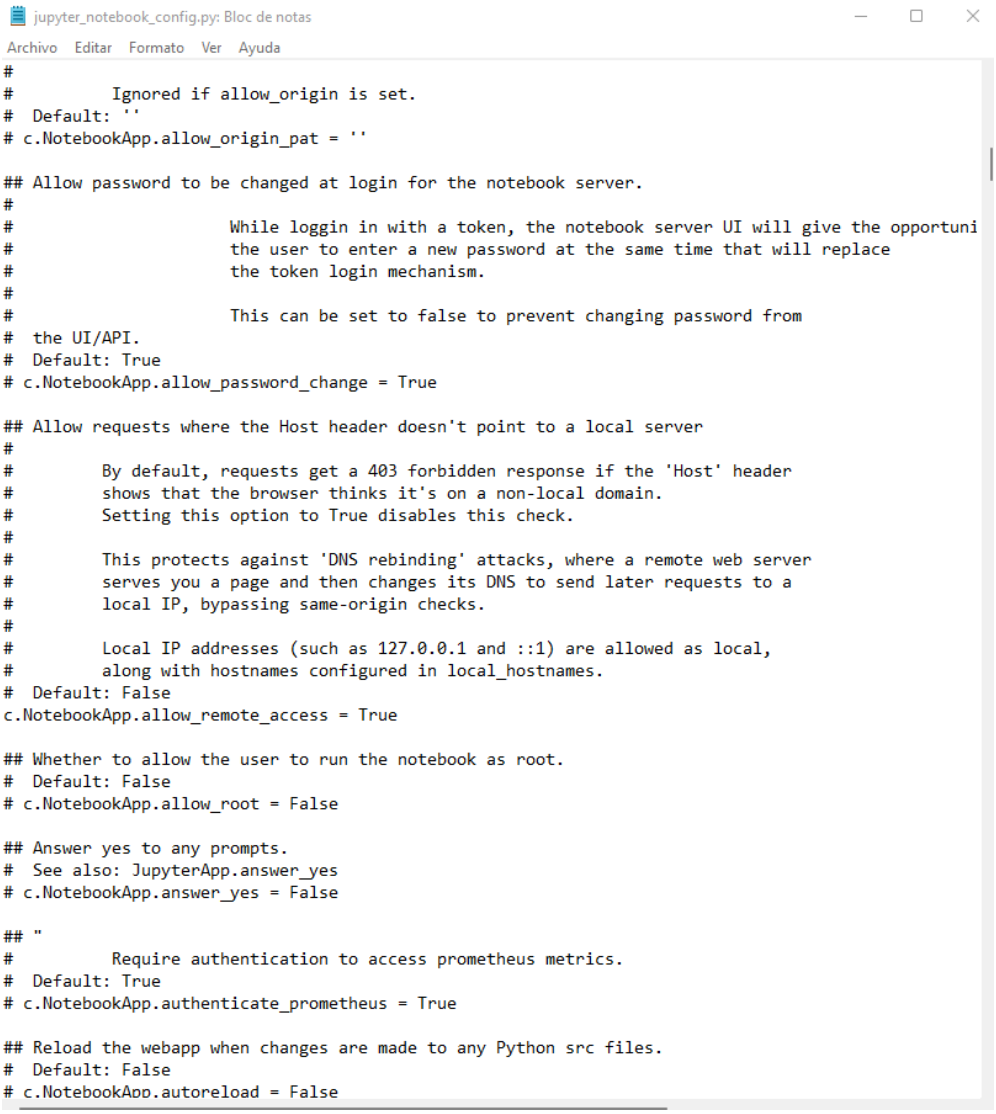

Ilustración 9: Archivo de configuración Jupyter Notebook

En el archivo de configuración mostrados en la ilustración 9 se puede modificar el puerto por el cual se provee el servicio, las direcciones ip permitidas, si se lanza el servicio automáticamente, etc.

Para acceder a la plataforma de *Jupyter Notebook* desde otro equipo ahora solo es necesario escribir la IP y el puerto del equipo que lanza el servicio de *Jupyter Notebook* en el buscador. Por este motivo es preferible establecer una ip estática en el equipo donde se iniciará el servicio, de esta manera no será necesario corroborar la ip antes de intentar conectarse a J*upyter Notebook*. Hasta este punto ya se puede ofrecer el servicio de *Jupyter Notebook* a una LAN, el siguiente paso para hacer operativa la mesa prototipo de resonancia natural es convertir el programa actual del ESP32 con todas sus funciones a lenguaje Python, ya que actualmente trabaja con la *IDE* de Arduino.

3.4 RUTINAS DE CONEXIÓN, CONTROL E INTERROGACIÓN

Al abrir J*upyter Notebook*, buscar la opción de nuevo y escoger la opción “Microython - USB” abrirá una pantalla que nos permitirá escribir rutinas de código que podremos correr individualmente, el lenguaje Python a diferencia del lenguaje C, no se “compila” como tal, sino que se va “interpretando línea por línea” por lo que se corre más rápidamente. A continuación, describiré cada uno de los segmentos de código utilizados para el control de la mesa prototipo.

3.4.1 Conexión al ESP32 mediante conexión serial.

```
%serialconnect to --port=COM3 -- baud=11520
```

En esta línea permite establecer conexión con el microcontrolador, lo importante es conocer previamente el puerto serial por el cual se está conectando el ESP32 la computadora. Esto se puede hacer mediante el administrador de dispositivos de Windows, pero lo más común es que sea COM4 o COM3 [18].

3.4.2 Desconexión del ESP32.

```
%disconnect to --port=COM3 -- baud=11520
```

Esta rutina de desconexión sirve para terminar procesos, desconectar el microcontrolador en caso que otro programa quiera tener acceso a él, o para reiniciar el microcontrolador en algunos casos. Ademas debe utilizarse cada vez que otro usuario diferente al que está conectado quiera utilizar la mesa de pruebas.

3.4.3 Control de motor y acelerómetro

```
import _thread as th

import time
from machine import Pin
from machine import ADC
from machine import SoftI2C
from machine import sleep
import mpu6050

control_motor = True
acelerometro = True
i2c = SoftI2C(scl=Pin(22), sda=Pin(21))
mpu= mpu6050.accel(i2c)

datos=[]
hz = 10
t_trabajo = 10
control1 = 20*t_trabajo

derecha = Pin(12, Pin.OUT)
izquierda = Pin(13, Pin.OUT)

def frecuencia(delay):
     while control_motor:
       derecha.value(1)
       time.sleep(delay)
       derecha.value(0)
       izquierda.value(1)
       time.sleep(delay)
       izquierda.value(0)


def medicion():
    for i in range(control1):
        time.sleep(0.05)
        mpu.get_values()
        print(mpu.get_values())

th.start_new_thread(frecuencia, (1/hz,))
th.start_new_thread(medicion, ())

time.sleep(t_trabajo)

control_motor = False
```

Esta rutina sirve para controlar el sentido del movimiento del motor, la frecuencia a la que oscila la mesa prototipo y el tiempo que durarán las mediciones, además permite al acelerómetro medir las aceleraciones a las cuales está sometida la mesa. La ventaja de poder editar el código al mismo tiempo que se va probando cómo trabaja la mesa es que se pueden editar los parámetros directamente creando así un movimiento más preciso para

simular el fenómeno que estamos estudiando, la variable hz del código simula la frecuencia a la cual oscila la mesa, mientras que la variable t_trabajo simula el tiempo de duración del movimiento y del muestreo de las mediciones.

Actualmente la mesa prototipo cuenta con 3 estructuras, al realizar pruebas variando los parámetros mencionados se encontró que la estructura más alta entra en resonancia a una frecuencia de 10Hz, la segunda estructura a una frecuencia de 12.5Hz y la estructura más pequeña a una frecuencia de 14.7Hz.

Además, el código permite la utilización de ambos núcleos del ESP32, uno el cual se encarga únicamente del movimiento del motor, y el otro que se encarga de guardar las mediciones del acelerómetro, ambos procesos deben estar enlazados para que puedan funcionar bien, sin embargo, a nivel de programación son 2 eventos distintos que gracias a los dos núcleos del ESP32 pueden funcionar simultáneamente. [19]

3.4.4 Reset

```
import machine
machine.reset()
reset
```

Esta rutina sirve para superar errores sin causa aparente, después de utilizar esta rutina será necesario volver a utilizar la rutina de conexión mediante el puerto serial. [20]

3.4.5 Almacenamiento de datos y control de mesa

```
import _thread as th
import time
from machine import Pin
from machine import ADC
from machine import SoftI2C
from machine import sleep
import mpu6050
import network
import urequests
import esp
import gc

esp.osdebug(None)
control_motor = True
acelerometro = True
gc.collect()
```

```
ssid = 'CLARO1_B7E807'
password = '789s4hnwLH'
api_key = 'cOzvUKVxlq56y3S06vT7we'

station = network.WLAN(network.STA_IF)
station.active(True)
station.connect(ssid, password)
while station.isconnected() == False:
    pass

print('Connection successful')
print(station.ifconfig())

i2c = SoftI2C(scl=Pin(22), sda=Pin(21))
mpu= mpu6050.accel(i2c)

datos=[]
hz= 0.5
t_trabajo = 3
control1 = 20*t_trabajo

derecha = Pin(12, Pin.OUT)
izquierda = Pin(13, Pin.OUT)

def frecuencia(delay):
     while control_motor:
       derecha.value(1)
       time.sleep(delay)
       derecha.value(0)
       izquierda.value(1)
       time.sleep(delay)
       izquierda.value(0)

def medicion():
    for i in range(control1):
        time.sleep(0.05)
        mpu.get_values()
        datos.append(mpu.get_values()*(9.81/16384.0))

th.start_new_thread(frecuencia, (hz,))
th.start_new_thread(medicion, ())

time.sleep(t_trabajo)

control_motor = False

try:
    medicion = datos

    sensor_readings = {'value1':medicion }

    request_headers = {'Content-Type': 'application/json'}
```

```
    request =
urequests.post('https://maker.ifttt.com/trigger/ESP32/with/key/' +
api_key , json=sensor_readings, headers=request_headers)
    print(request.text)
    request.close()
except OSError as e:

    print('Failed to read/publish sensor readings.')
```

Este código permite controlar totalmente la mesa prototipo y al mismo tiempo almacenar los datos en G*oogle Drive*, de igual manera las únicas variables que se modifican son hz y t_trabajo como se explicó anteriormente.

3.5 ETAPA DE CONEXIÓN A GOOGLE CLOUD

Para lograr conectar dos o más redes *LAN* diferentes utilizamos *Google Cloud*, en esta plataforma es necesario registrarnos con nuestro correo electrónico para poder utilizarla y luego configurar el firewall para que permita la conexión de otras redes y puertos. Posteriormente se crea una instancia virtual en la cual utilizaremos Ubuntu ya que es una de las virtual machine que están habilitadas para ser utilizadas de manera gratuita en G*oogle Cloud*.

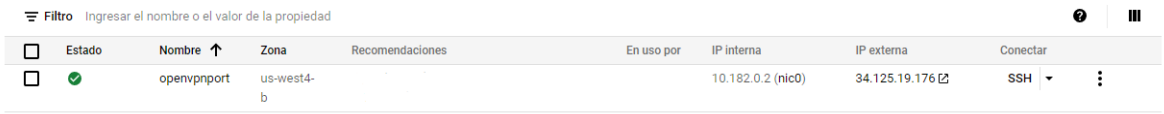

Ilustración 10: Instancia de máquina virtual, Google Cloud

Como se ve en la ilustración 10 *Google Cloud* asigna una IP pública y una IP privada a la instancia virtual, esta instancia virtual será utilizada como servidor para poder comunicar distintas redes *LAN*, de esta manera solo será necesario establecer una conexión mediante *VPN* a este servidor. Ya que el proyecto está pensado para ser utilizado en la escuela de Ingeniería Eléctrica y ahí no se tiene acceso a la configuración de los routers de la red es necesario apoyarnos de una aplicación externa para establecer la conexión *VPN*, esta vez se utilizó O*penVPN*, el cual debe ser instalado en la virtual machine de G*oogle Cloud* y ahí mismo se agregarán y generan los archivos de los usuarios que tendrán acceso al servidor.

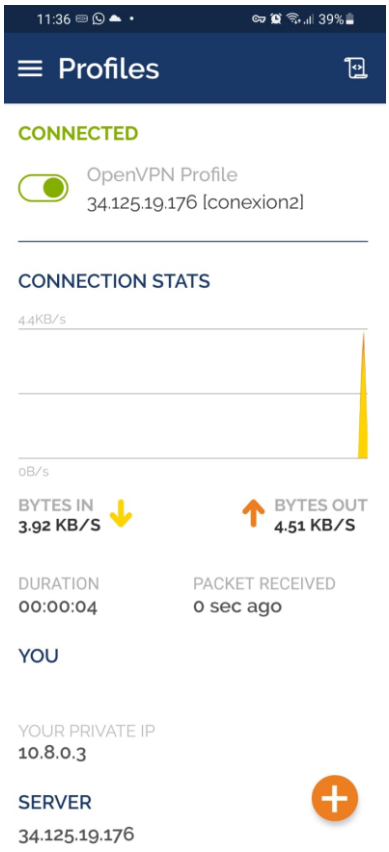

Ilustración 11: Open VPN conectada al servidor en Google Cloud

Otra ventaja es que esta aplicación está disponible para Android, Windows y MAC por lo que se puede controlar la mesa prototipo desde cualquiera de estos dispositivos, para poder utilizar este programa es necesario transferir los archivos generados por *OpenVPN* en la virtual machine a los usuarios, y estos se cargan en *OpenVPN* instalado en los equipos remotos.

Una vez habilitada la conexión, así como aparece en la ilustración 11, solo es necesario escribir la IP del equipo que lanza el servicio (que puede ser cualquiera de los equipos conectados al servidor) y seleccionar el puerto en el buscador de preferencia.

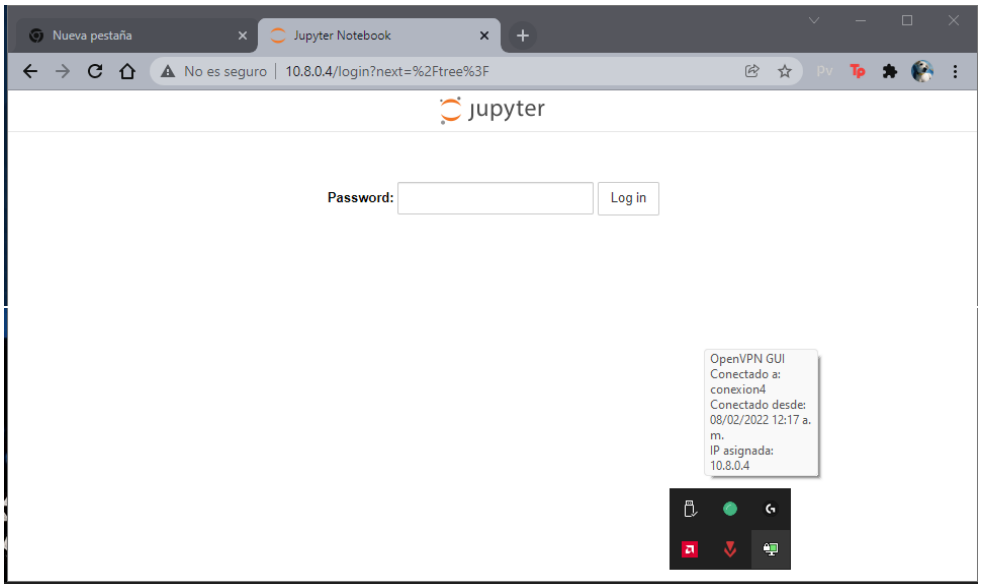

Ilustración 12: Jupyter Notebook conectado a la red del servidor en Google Cloud mediante Open VPN

En este caso se ha configurado una contraseña tal y como aparece en la ilustración 12 para acceder al servicio de *Jupyter Notebook*, esto como una medida adicional de seguridad para controlar el acceso de los usuarios. Una vez hecho esto solo se ingresa al programador de "MicroPython-USB" en el botón de "new" y se pueden ejecutar las rutinas de código Python que se describieron anteriormente.

Para el almacenamiento de datos se utiliza *Google Drive*, para conectarnos a este servicio nos auxiliamos de otro servicio web llamado *IFTTT*, se utiliza este método porque en M*icropython* no existe una librería que pueda conectar directamente estos dos servicios, y ya que IFTTT sólo necesita una *reques*t para realizar una acción resulta ser muy práctica para subir datos a *Google Drive* [21].

Al realizar la configuración necesaria que se explica en la "guía de instalación" *IFTTT* te asigna una dirección web única a la cual se debe enviar una solicitud en donde se incluye la o las variables con los datos que se desean guardan en *Google Drive*, en este caso y por motivos de orden se decidió guardar los datos en un documento de Google dentro de *Google Drive*, sin embargo, existe la opción para no hacer esto y guardarlo directamente en un bloc de notas dentro de *Google Drive*. La dirección de Google se puede escoger a conveniencia y se recomienda utilizar una dirección específicamente para esto.

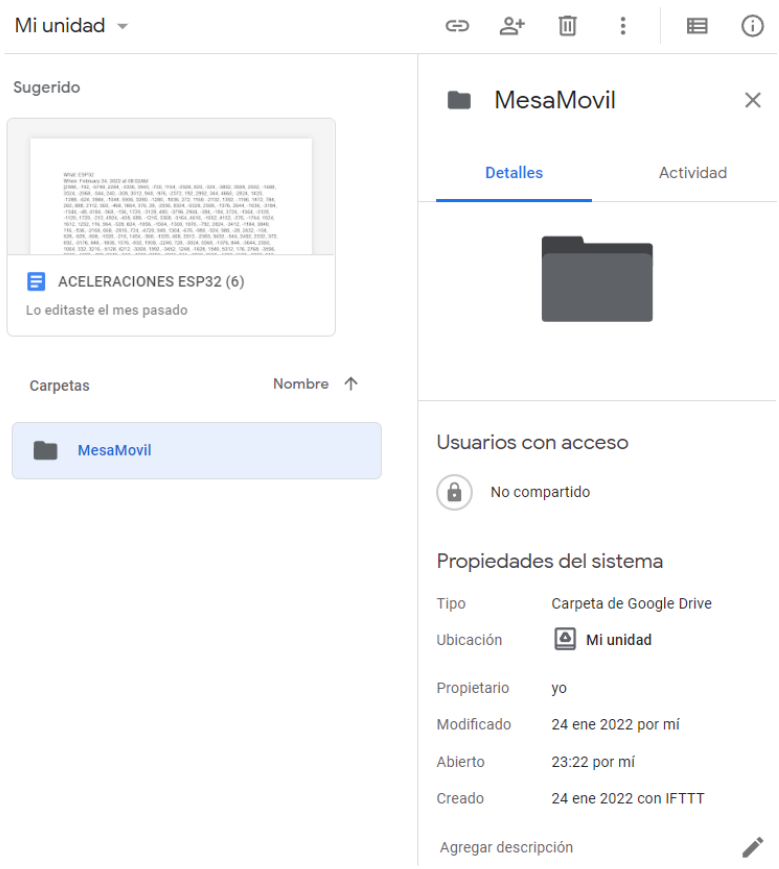

Ilustración 13: Google Drive de almacenamiento de mediciones.

Como se aprecia a observar en la ilustración 13 crear carpetas para ordenar mejor los documentos con los datos, de esta manera se pueden tener diferentes mediciones de distintos equipos y tener la información ordenada, además el documento nuevo se guarda como copia del original, por lo que es posible tener datos históricos sin preocuparnos por guardar los datos cada vez que realicemos una prueba.

Una vez realizadas todas estas modificaciones ya es posible operar la mesa móvil, por lo que su funcionamiento y resultados serán abordados en el siguiente capitulo.

CAPÍTULO 4

4.1 RESULTADOS

Ya que la mesa móvil está diseñada para observar los resultados de manera visual, se dificulta el presentar los resultados en un documento, sin embargo, se tratará de abordar de la mejor manera para poder explicar los resultados y las pruebas.

4.2 ESTRUCTURAS

Para las estructuras se utilizaron bloques de madera y varillas de soldadura de bronce, estas estructuras no fueron modificadas ya que permitían observar el fenómeno de resonancia fácilmente, por lo que se decidió mantenerlas como estaban inicialmente. La estructura utilizada está representada en la ilustración 14.

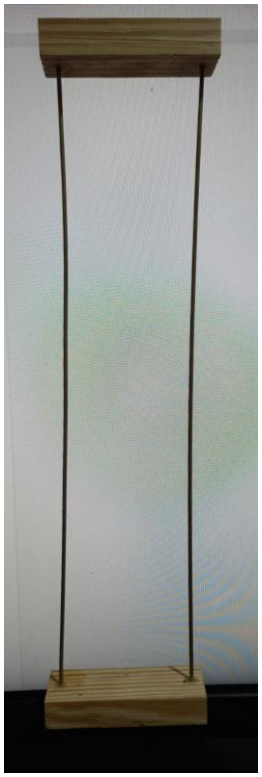

Ilustración 14: Estructura utilizada en mesa de pruebas

4.2 FRECUENCIA NATURAL DE CADA ESTRUCTURA

El objetivo final del trabajo de graduación es obtener una mesa móvil que sea capaz de hacer entrar en resonancia a las estructuras que se coloquen sobre ella, por lo que inicialmente se hicieron pruebas utilizando la mesa a distintas frecuencias hasta encontrar aquella en donde visualmente las aceleraciones de las estructuras sean máximas, esto es observable cuando la estructura se mueve a lo largo de un eje y la amplitud de su movimiento es la mayor posible, además es necesario hacer cambios pequeños en los valores de la frecuencia, ya que incluso variar en 1 o 2 Hz produce cambios significativos en el movimiento que se produce en las estructuras, como muestra de eso se presentan las siguientes imágenes de cada estructura entrando en resonancia, con su respectivas mediciones y dimensiones.

Para la estructura más alta cuyas dimensiones es de 25.9 cm el valor de su frecuencia natural es aproximadamente: 10Hz

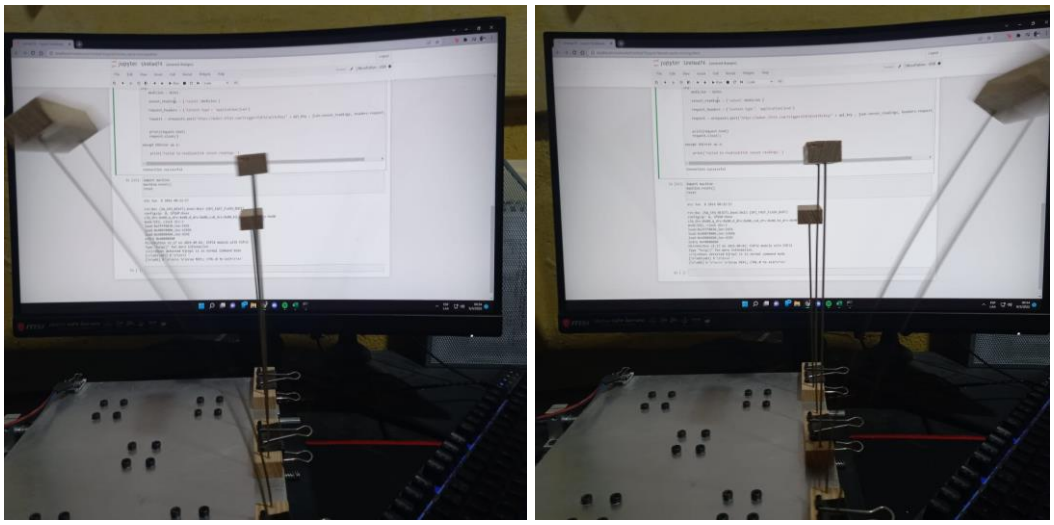

Ilustración 15: Estructura 1 en resonancia

Para la estructura mediana cuyas dimensiones es de 21.8 cm el valor de su frecuencia natural es de aproximadamente: 12.5Hz

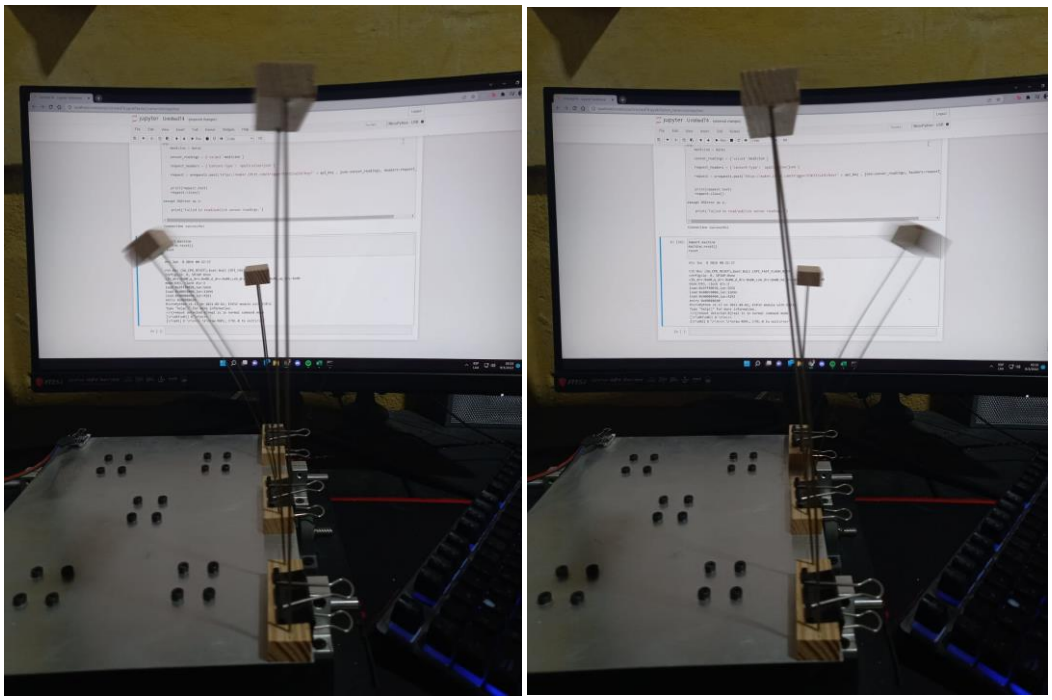

Ilustración 16: Estructura 2 en resonancia

Para la estructura más pequeña cuyas dimensiones es 17 cm el valor de su frecuencia natural es de aproximadamente: 14.7Hz

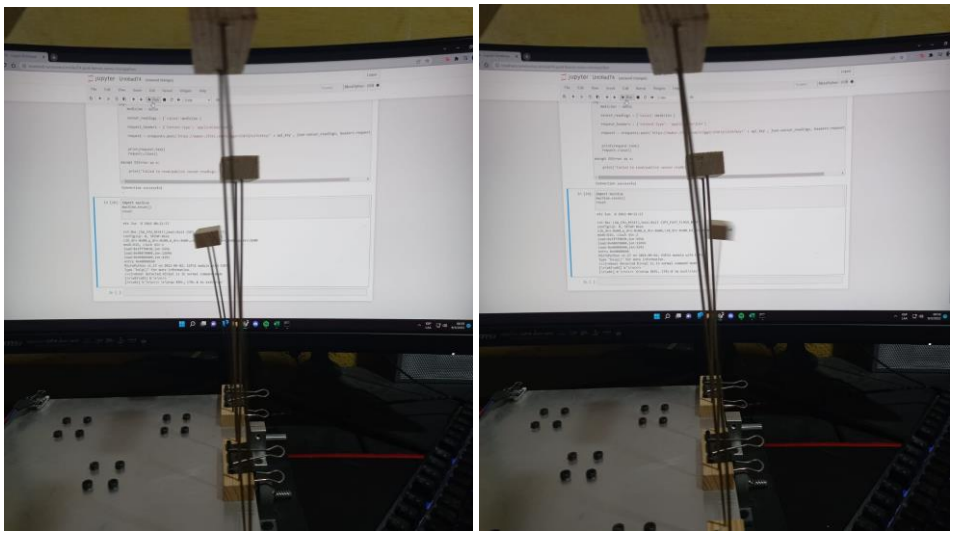

Ilustración 17: Estructura 3 en resonancia

Las mediciones del acelerómetro varían según la frecuencia en que se programe el equipo, por default está configurado a realizar mediciones durante un periodo de 10 segundos, lo cual cumple con el principio de muestreo de Nyquist, actualmente se hace un registro de 20 mediciones por segundo, lo cual al final de los 10 segundos da como resultado un

aproximado de 200 mediciones, lo suficiente como para muestrear de la manera correcta cualquier tipo de frecuencia que se pueda programar en la mesa de pruebas.

Para el análisis de los datos se utiliza de igual manera *Jupyter Notebook*, solo que esta vez con un notebook de Python 3 y no de M*icropython,* ya que la extensión de Micropython solo incluye funciones relacionadas al control de microcontroladores, para trasladar la información de un notebook a otro se hace copiando la información manualmente.

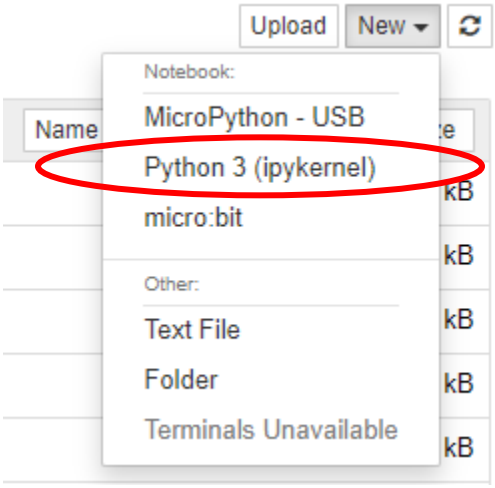

Ilustración 18: Notebooks instalados

Para facilitar el uso de los datos se recomienda copiarlos directamente de Google Drive, ya que este ya posee el formato adecuado para solo copiarlo y pegarlo, el código utilizado para graficar y presentar los datos es el siguiente:

```
import matplotlib.pyplot as plt
import numpy as np
from tabulate import tabulate
datos= []
tiempo2 = 0
tiempo = []
espacio = "        "
numero_datos = len(datos)
for i in range(numero_datos):
    tiempo1=round(1/20+tiempo2,2)
    tiempo.append(tiempo1)
    tiempo2=tiempo1
print ("{:<8} {:<15}".format('Tiempo[s] ','Medición [m/s]'))
for i in range(numero_datos):
    print ( tiempo[i], espacio, datos[i])
    plt.plot(tiempo,datos)
plt.show()
```

Este código incluye librerías que facilitan el tratamiento de la información, para presentarla gráficamente y tabulada, para que los resultados sean más compresibles para el usuario. Al utilizar los datos de una de las mediciones la salida del código anterior se ve reflejado en la ilustración 20.

Tiempo[s]	Medición [m/s]
0.05	-1.053809
0.1	-0.4263135
0.15	-0.5101392
0.2	2.48603
0.25	0.2227368
0.3	-1.499282
0.35	2.224973
0.4	-2.907554
0.45	0.02874023
0.5	0.7544312
0.55	-2.44771
0.6	1.908831
0.65	0.366438
0.7	-0.9915381
0.75	1.506467
0.8	1.029858
0.85	0.2921924
9.7	0.1149609
9.75	0.1317261
9.8	0.3161426

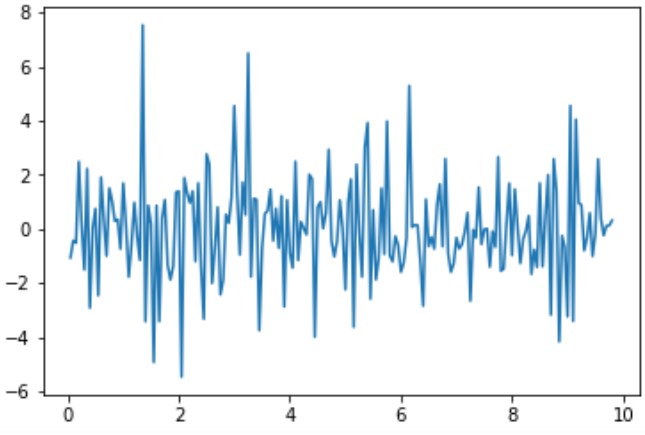

Ilustración 19: Notebook de tratamiento de resultados.

La imagen ha sido recortada por motivos de espacio, pero en ella se deberían observar todos los datos registrados durante la medición con su respectivo tiempo, además en la gráfica de los datos podemos observar distintos picos que no corresponden totalmente a la onda esperada, ya que por la naturaleza del movimiento este debería verse como una onda senoidal.

Finalmente se presenta un esquema plasmado en la ilustración 20 en donde resume el funcionamiento lógico de la mesa móvil, desde su etapa de control hasta su etapa de conexión, en donde se muestra la relación que existe entra cada una de sus partes.

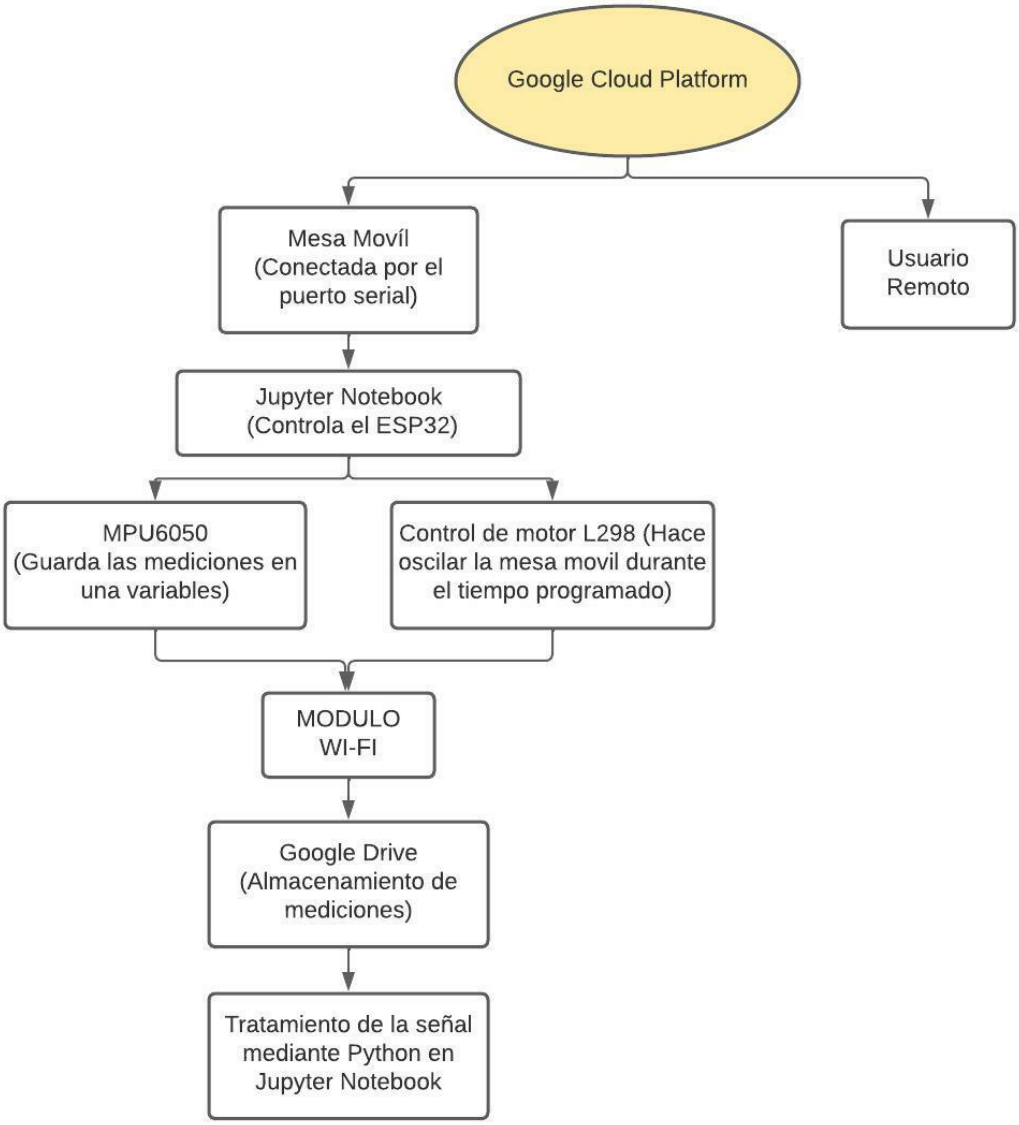

Ilustración 20: Esquema de funcionamiento del sistema de conexión y control de la mesa móvil

En resumen, Google Cloud sirve como servidor para la creación de redes privadas virtuales, una vez establecida la conexión entre distintos usuarios ya es posible ejecutar el servicio de Jupyter Notebook y conectarse a la mesa móvil. Una vez lanzado el servicio de Jupyter Notebook es necesario poder entender y modificar los códigos propuestos en la sección 3.4, de esta manera se podrá tener control absoluto sobre la mesa de pruebas, sacando todo el provecho posible a las características del ESP32.

Posteriormente a realizar las mediciones y guardarlas mediante Google Drive estos quedan disponibles para el tratamiento de ellos, actualmente el programa propuesto se ha escrito en Python y se utiliza la misma plataforma de Jupyter Notebook para ser utilizado.

CAPÍTULO 5

5.1. LÍNEA FUTURA

Este tema presenta una gran cantidad de oportunidades de mejora, estas se dividirán separándolas según su área de aplicación:

1. **Comunicación:** Con el objetivo de evitar las limitantes descritas en el documento por Google Cloud, se podría solicitar el acceso a un puerto de la IP pública de la universidad, de esta manera se podrían configurar los routers para poder tener acceso al control de la mesa mediante J*upyter Note*book desde cualquier red remota.
2. **Hardware:** Ya que el efecto de la resonancia depende en gran medida de la visualización en tiempo real del efecto, es necesario instalar una cámara permanente en la mesa móvil, la cual permita grabar o transmitir la imagen mientras se utiliza la mesa.
3. **Mediciones:** Con el propósito de comprobar la teoría acerca de la resonancia sería conveniente la instalación de acelerómetros en las estructuras, para esto es necesario diseñar estructuras en donde el cable y la masa del acelerómetro afecten en la menor medida posible al movimiento. Al tener ambas mediciones será posible comparar las magnitudes y comprobar la teoría detrás de esto.
4. **Control:** *Jupyter Notebook* permite la manipulación de la mesa de forma bastante efectiva, por lo que el diseñar nuevos notebooks o programas para realizar prácticas con la mesa móvil es crucial para poder darle continuidad a este proyecto.
5. **Estructuras:** Actualmente se utilizan estructuras bastante sencillas, por lo que es recomendable diseñar nuevas estructuras que incluso se asemejan más a los edificios modernos, así podremos interpretar y darles más valor a los resultados de las mediciones.
6. **Amortiguamiento:** Se pueden implementar distintas practicas en donde se incluyan otro tipo de estructuras y movimientos, uno de esas prácticas podría ser acerca de amortiguadores basados en sistemas reales actuales para la disipación de la resonancia.

CONCLUSIONES

1. El prototipo de conexión para la mesa móvil es funcional, por lo que ahora se podrá utilizar de forma inalámbrica incluso cuando los usuarios estén conectados en distintas redes, esto será realmente útil para la realización de prácticas de laboratorio de manera virtual, o para usar más eficientemente el equipo dentro del laboratorio, superando así la limitación de necesitar un equipo por grupo de estudiantes.

2. Debido a que la VPN que se utiliza es a través de un servicio de Google será necesario realizarle mantenimiento a la red, aproximadamente cada 3 meses, ya que este es el tiempo en que Google Cloud deja de funcionar de forma gratuita y será necesario levantar de nuevo el servició y la instancia virtual.

3. A pesar que la mesa ya es funcional un existen grandes oportunidades de mejora para la mesa móvil, ya que el objetivo final es profundizar y facilitar el aprendizaje del tratamiento digital de la señal en señales sísmicas, por lo que se presentaron varias oportunidades de mejora para realizarse a futuro.

4. Con el desarrollo de este trabajo de graduación ahora se puede controlar la mesa móvil directamente con código el Python, lo cual incrementa la precisión con la cual se puede utilizar. Además, al poder controlarse mediante internet aumenta en gran medida la accesibilidad que existe para utilizarla.

BIBLIOGRAFÍA

[1] César Salguero García y. Adrian Mira Trigueros, «PROTOTIPO DE MESA DE PRUEBAS, PARA DEMOSTRAR RESONANCIA EN ESTRUCTURAS,» San Salvador, 2019.

[2] Adrian Mira Trigueros y César Salguero García, «MESA DE PRUEBAS PARA EL ESTUDIO DE RESONANCIA EN ESTRUCTURAS,» San Salvador, 2019.

[3] Adrian Trigueros, Diseño y construccion de un simulador de resonancia en estructuras, San Salvador, 2021.

[4] Jupyter Team, «https://jupyter.org.,» 2015. [En línea]. Available: https://jupyter-notebook.readthedocs.io/en/latest/notebook.html. [Último acceso: 02 febrero 2022].

[5] «https://micropython.org/,» 2018. [En línea]. Available: https://micropython.org/. [Último acceso: abril 2022].

[6] «randomnerdtutorials.com,» RandomNerdTutorials, 2022. [En línea]. Available: https://randomnerdtutorials.com/install-upycraft-ide-windows-pc-instructions/. [Último acceso: abril 2022].

[7] Google Team, «cloud.google DOCUMENTACION DE COMPUTE ENGINE,» [En línea]. Available: https://cloud.google.com/compute/docs/?_ga=2.49743590.-1962908106.1637827107&_gac=1.191295704.1648652889.Cj0KCQjw_4-SBhCgARIsAAlegrV7xC2YdkaX-SxD3sS20w3L_yyEHYuKNVwZHckoOlEH1_9tiFiMSEAaAk2aEALw_wcB. [Último acceso: 06 febrero 2022].

[8] Open VPN Team, «OpenVpn.net,» 2022. [En línea]. Available: https://openvpn.net/access-server/. [Último acceso: febrero 2022].

[9] José Antonio Peralta, «El fenómeno de la resonancia,» Departamento de Fisica, Escuela Superior de Física y Matematicas Instituto Poltécnico Nacional, México D.F., 2009.

[10] «naylampmechatronics,» [En línea]. Available: https://naylampmechatronics.com/blog/45_tutorial-mpu6050-acelerometro-y-giroscopio.html. [Último acceso: abril 2022].

[11] William Schafer, Tratamiento de señales en tiempo discreto, PEARSON, 2011.

[12] E. Ambitiously, «Adquirir una Señal Analógica: Ancho de Banda, Teorema de Muestreo de Nyquist y Aliasing,» 5 marzo 2019. [En línea]. Available: https://www.ni.com/es-cr/innovations/white-papers/06/acquiring-an-analog-signal--bandwidth--nyquist-sampling-theorem-.html. [Último acceso: febrero 2022].

[13] J. Monge, «Biela-manivela,» Conselleria d'Educació Generalitat Valenciana, [En línea]. Available: https://www.edu.xunta.gal/espazoAbalar/sites/espazoAbalar/files/datos/1464947673/contido/53_bielamanivela.html. [Último acceso: febrero 2022].

[14] «Amazon,» [En línea]. Available: https://www.amazon.com/-/es/Ideaker-horizontal-19-685-0-079-acoplamiento/dp/B01LQ9OG80. [Último acceso: febrero 2022].

[15] RETIA, «How to Control Electronics from a Browser Using MicroPython in Jupyter Notebook,» 20 julio 2020. [En línea]. Available: https://null-byte.wonderhowto.com/how-to/control-electronics-from-browser-using-micropython-jupyter-notebook-0236726/. [Último acceso: octubre 2021].

[16] K. Paul, «stackoverflow.com,» enero 2019. [En línea]. Available: https://stackoverflow.com/questions/42848130/why-i-cant-access-remote-jupyter-notebook-server/54063685#54063685. [Último acceso: pctubre 2021].

[17] J. VALIENTE, «cetatech.ceta-ciemat,» mayo 2016. [En línea]. Available: https://cetatech.ceta-ciemat.es/2016/05/instalalacion-de-un-entorno-jupyterhub/. [Último acceso: noviembre 2021].

[18] M. Rovai, «towardsdatascience,» junio 2018. [En línea]. Available: https://towardsdatascience.com/micropython-on-esp-using-jupyter-6f366ff5ed9. [Último acceso: noviembre 2021].

[19] Microcontrollers Lab, «microcontrollerslab.com,» junio 2021. [En línea]. Available: https://microcontrollerslab.com/micropython-mpu-6050-esp32-esp8266/. [Último acceso: noviembre 2021].

[20] Micropython contributors, «docs.micropython.org,» Noviembre 2016. [En línea]. Available: https://docs.micropython.org/en/v1.8.6/wipy/wipy/tutorial/reset.html. [Último acceso: Noviembre 2021].

[21] «Microcontrollerslab.com,» [En línea]. Available: https://microcontrollerslab.com/micropython-esp32-esp8266-send-sensor-readings-via-email-ifttt/. [Último acceso: octubre 2021].

ANEXOS

ANEXO 1: GUIA DE INSTALACIÓN JUPYTER NOTEBOOK

JUPYTER NOTEBOOK

Instalación de driver para ESP32 en Windows: este driver es necesario para que la computadora reconozca el dispositivo, por lo que se puede descargar de la siguiente pagina web: https://docs.espressif.com/projects/esp-idf/en/latest/esp32/get-started/establish-serial-connection.html

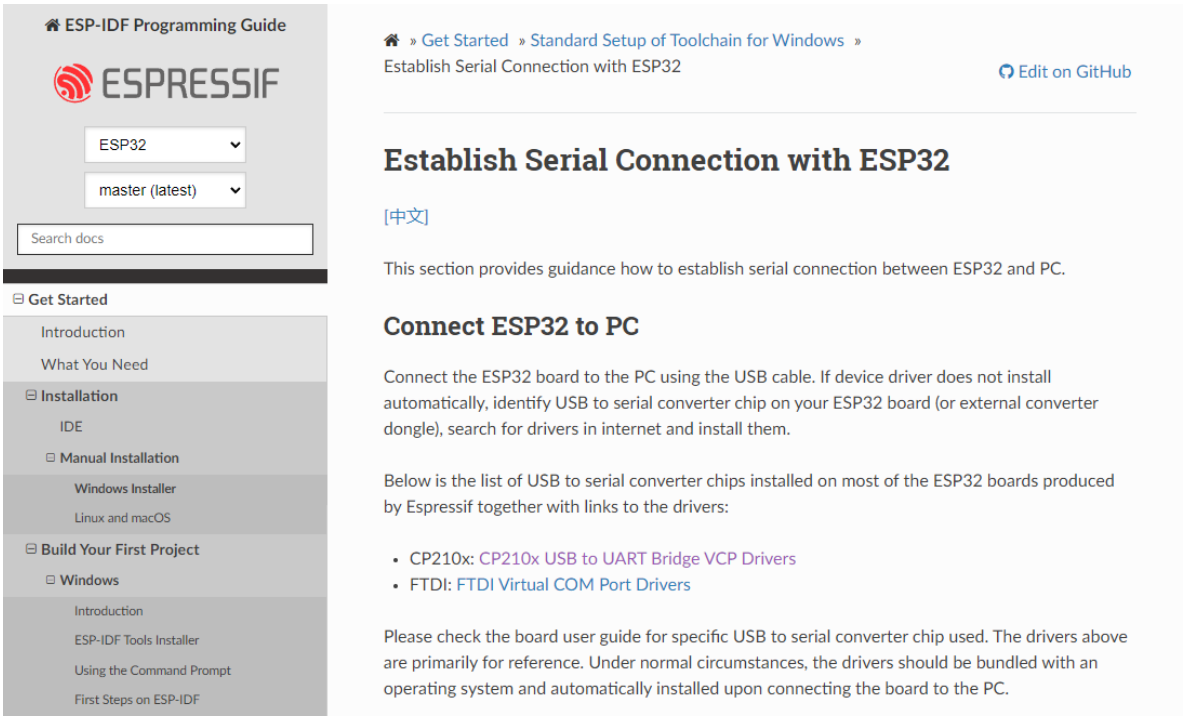

Ilustración 21: Sitio Web para descargar controlador CP210x para Windows.

Dependiendo del ESP32 se podría ocupar el driver “CP210x” o “FTDI”, estos drivers permiten la comunicación serial con el puerto USB, basta con descargarlo e instalarlo en la computadora para que este empiece a funcionar y permita reconocer el ESP32.

Identificación de puerto: Esto se hace mediante el administrador de dispositivos, para encontrar esto basta con escribir “administrador de dispositivos” en la barra de Windows (variara dependiendo del idioma de Windows).

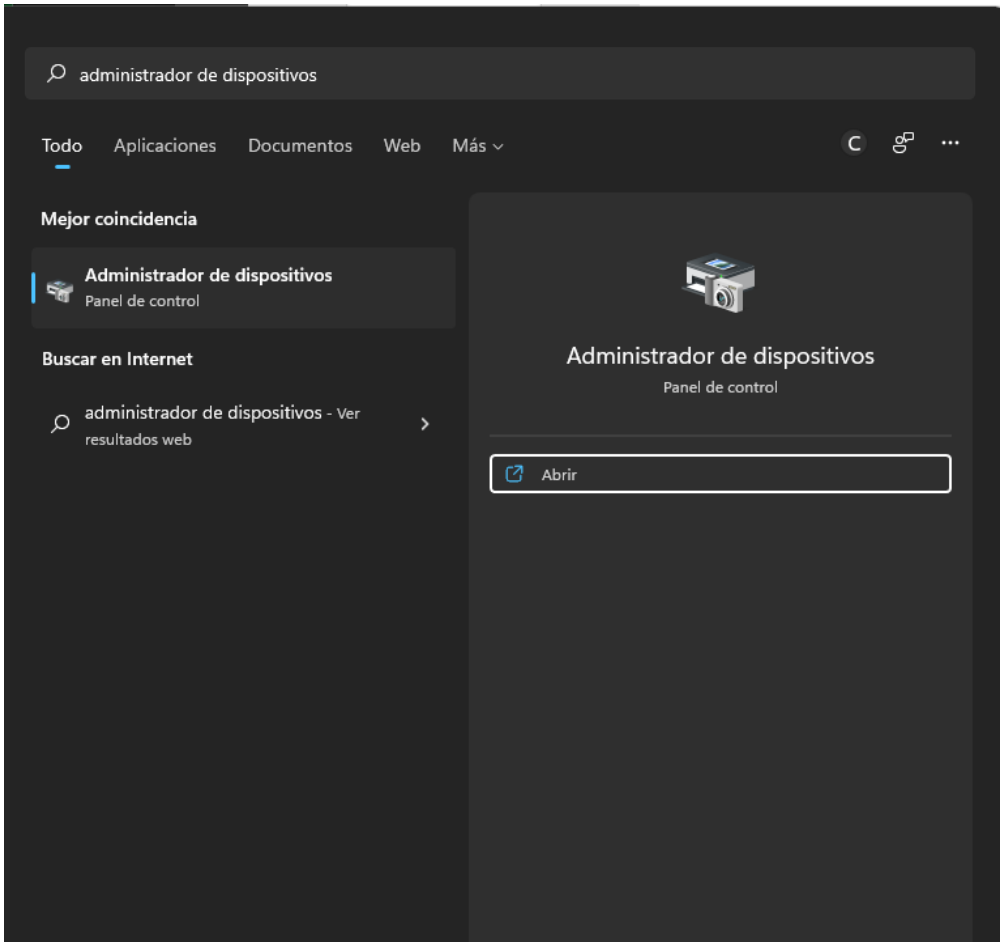

Ilustración 22: Administrador de dispositivos en buscador de Windows.

Una vez dentro seleccionamos el apartado de “Puertos” en el cual al expandirlo podremos identificar si el microcontrolador ha sido reconocido, y el puerto serial al cual está conectado.

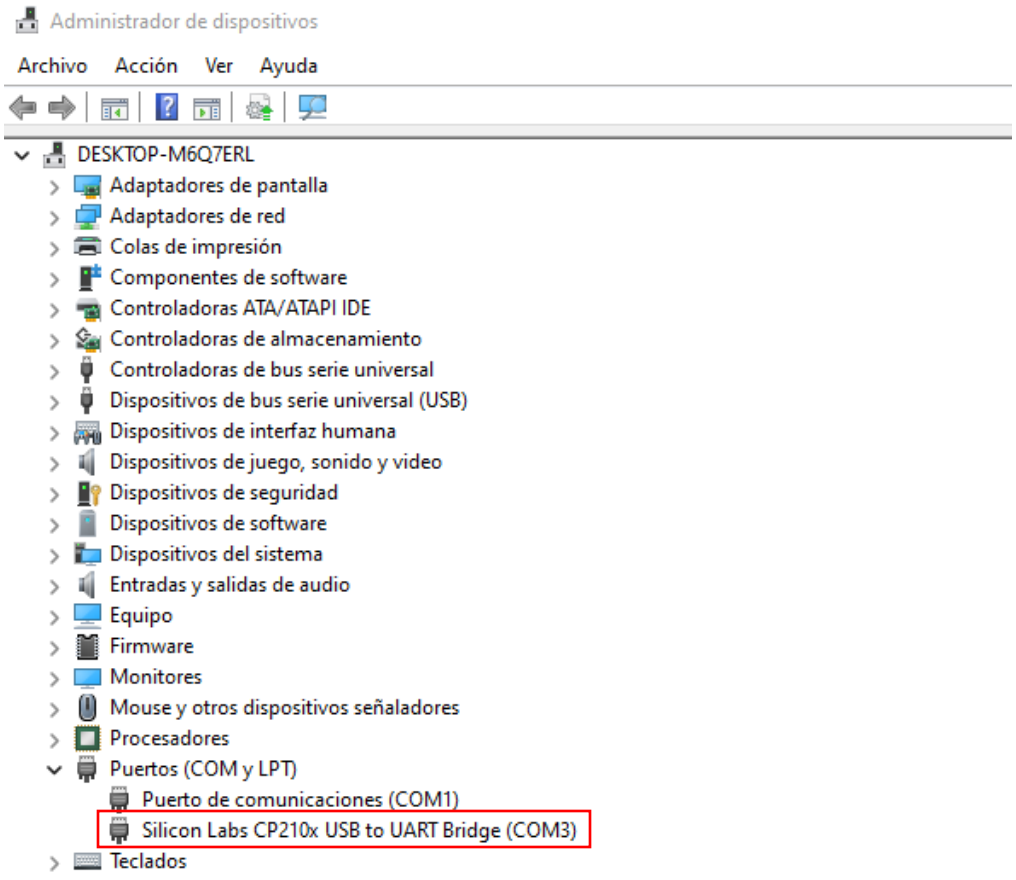

Ilustración 23: Identificación de puerto utilizado por ESP32

Instalación de firmware para ESP32: El firmware puede descargarse directamente de la página oficial de Micropython en el siguiente link:

https://micropython.org/download/esp32/ el cual además del firmware también se encuentran las instrucciones de como realizarlo, sin embargo, es necesario realizar cambios en esas líneas de comando para que estas funcionen en Windows.

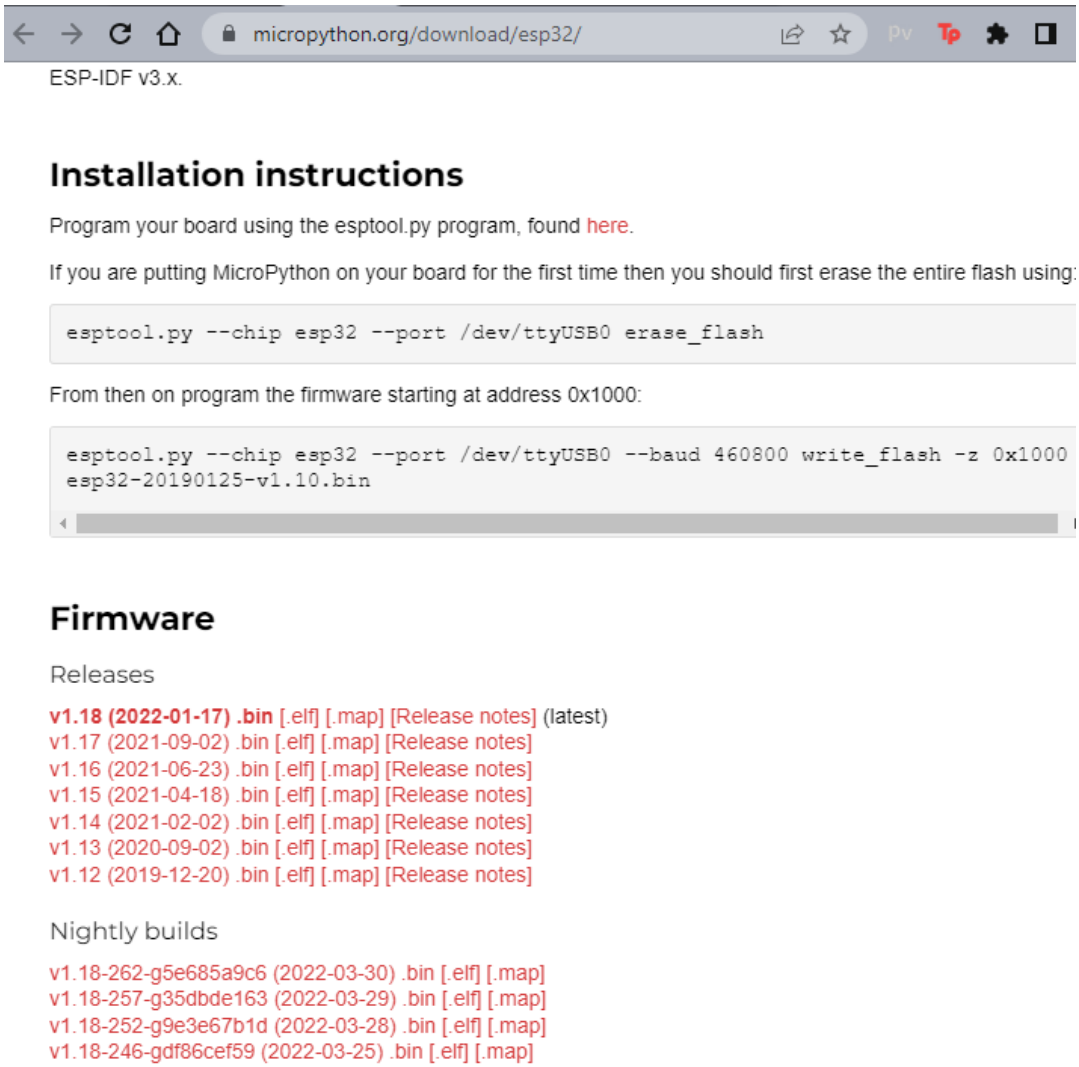

Ilustración 24:Sitio web para descargar firmware para micropython en ESP32

Es recomendable utilizar la última versión estable que existe para el microcontrolador, al momento de tomar la captura esta sería la versión “v1.18 (2022-01-17). Antes de instalar el nuevo firmware es necesario tener instalados en la computadora la herramienta esptool y Python 3, de lo contrario las líneas de comando que se presenten a continuación no podrán ser ejecutadas. Para instalar Python basta con visitar su página oficina www.python.org y descargar la versión más reciente como se muestra en la siguiente imagen.

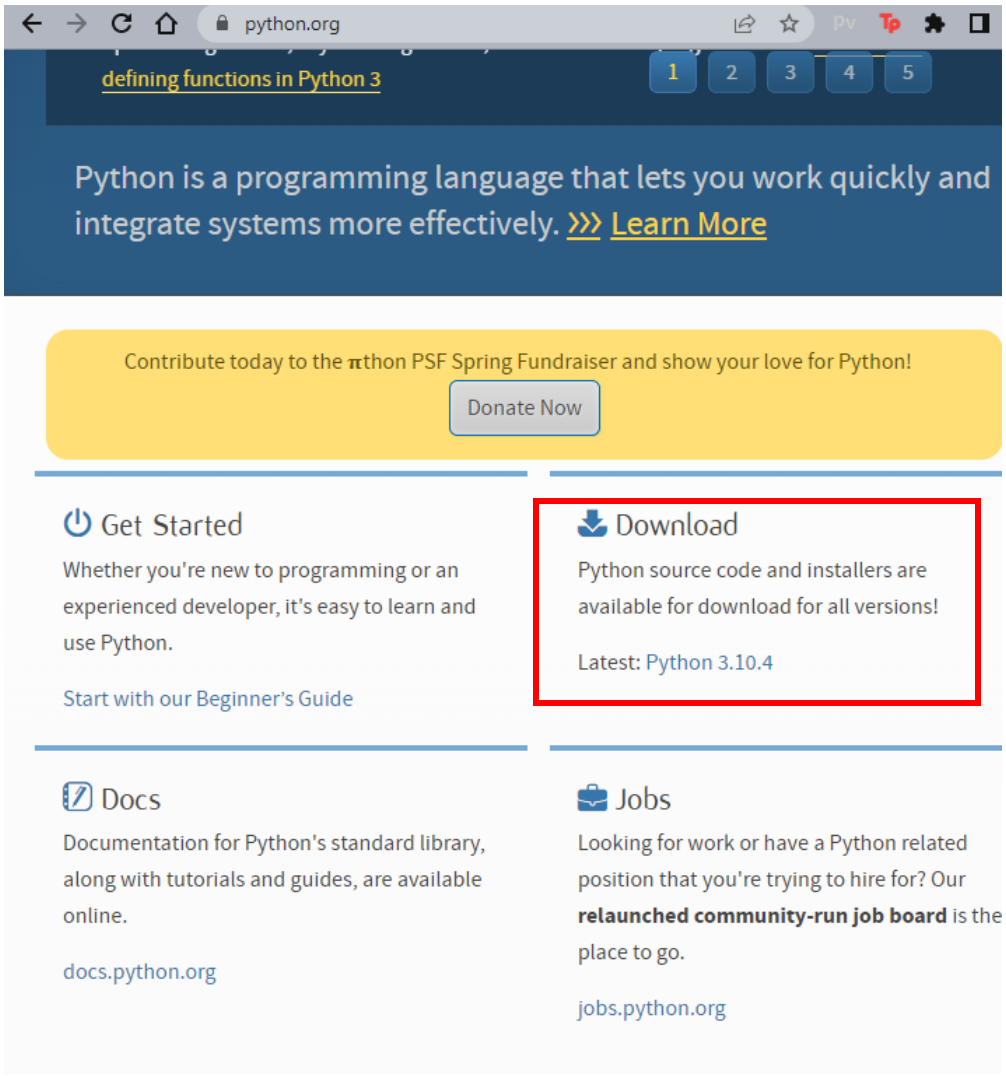

Ilustración 25: Sitio web para la descarga de Python para Windows.

Posteriormente se procede a instalarlo siguiendo el asistente de instalación y en donde debemos asegurarnos de marcar la casilla “add Python 3.7 to PATH”.

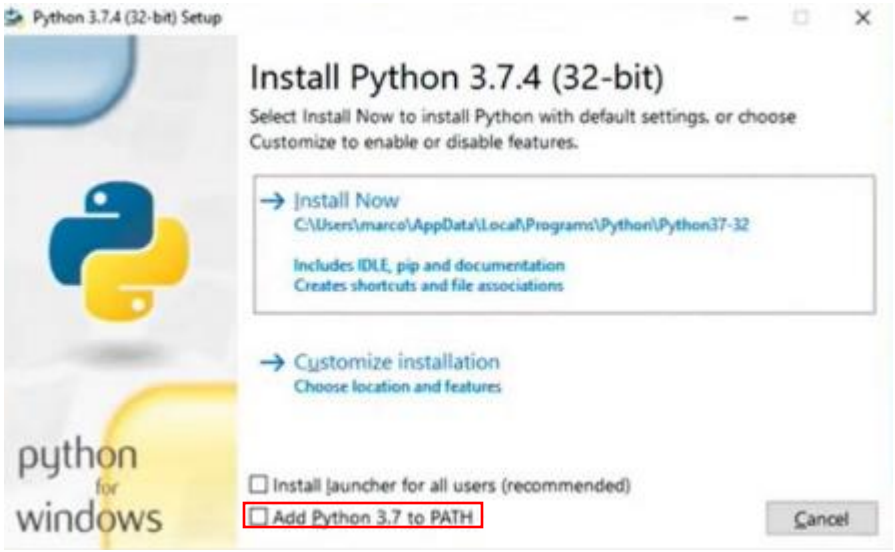

Ilustración 26: Menú de instalación para Python en Windows

Una vez finalizado esto procedemos a instalar la herramienta de *esptool,* lo cual se realiza con la siguiente línea de comando directamente en CMD.

Python -m pip install esptool

En caso de error es posible que Windows propiamente te sugiera una línea de comando para instalar esptool, en caso se ser asi es recomendable utilizarla. La línea de comando sugerida por Windows para la instalación de esptool en este caso es la siguiente:

C:\Users\cesar\AppData\Local\Programs\Python\Python310\python.exe -m pip install --upgrade pip

Este comando variará dependiendo del usuario ya que será necesario la ubicación de donde se instaló Python.

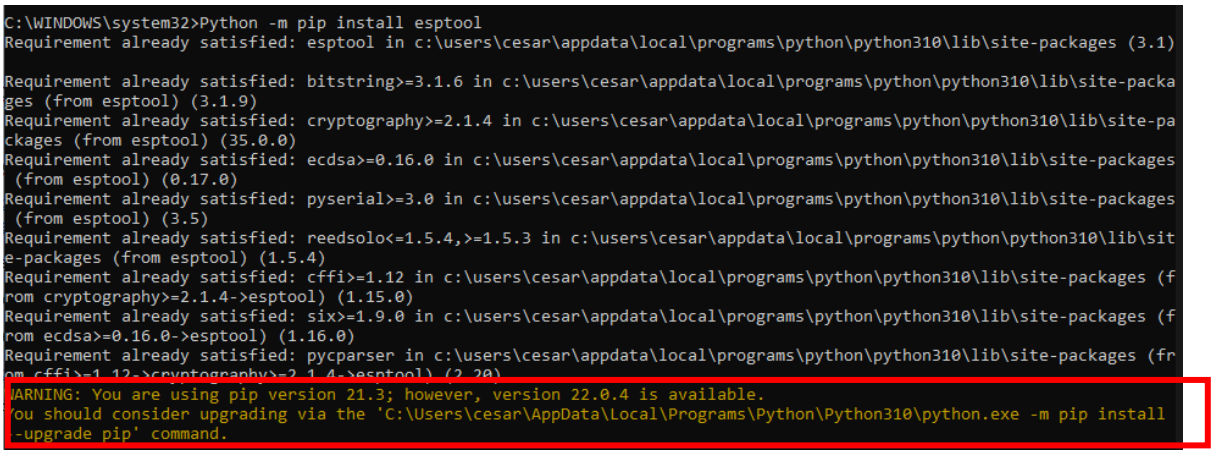

Ilustración 27: CMD ejecutando la linea de comando Python -m pip install esptool

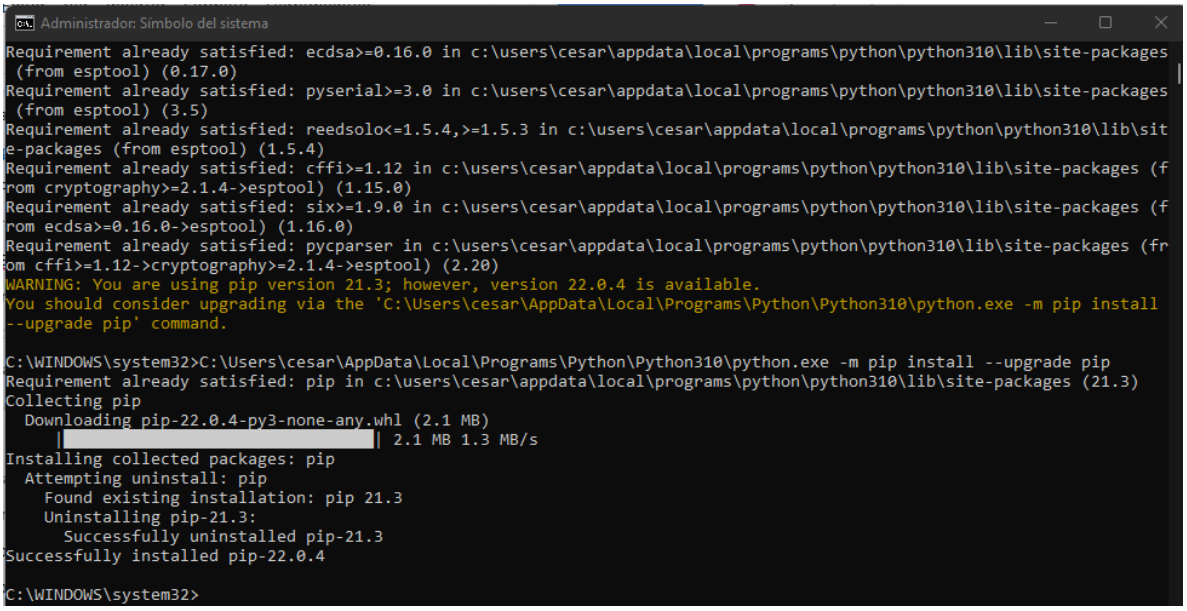

Ilustración 28: CMD ejecutando la línea de comandos según directorio

Para un microcontrolador ESP32 nuevo será necesario formatear la unidad, esto se hace para evitar cualquier problema que pudiese surgir al momento de la instalación del nuevo firmware. Para esto se utiliza la siguiente línea de comando.

esptool.py --port COM3 erase_flash

En algunos casos será necesario oprimir el interruptor llamado “boot” que viene incluido en el microcontrolador, o puede haber errores de no conexión como el mostrados en la siguiente imagen.

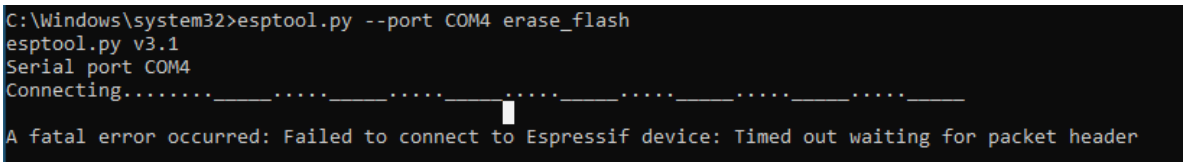

Ilustración 29: CMD fallando en la conexión con el ESP32

Sin embargo, si al momento de ejecutar la línea de comando mantenemos oprimido "boot" el proceso se llevará con normalidad.

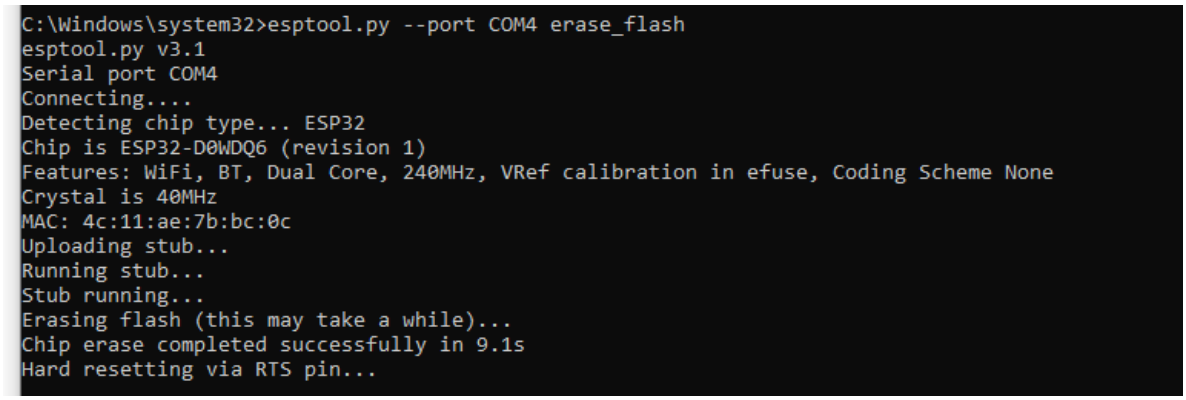

Ilustración 30: CMD ejecutando la línea de comando esptool.py --port COM4 erase_flash

El puerto escrito en la línea de comando dependerá del puerto serial que la computadora este utilizando para comunicarse con el ESP32, en la imagen anterior podremos observar cómo funciona, en la imagen aparece el puerto COM4 ya que ese se utilizo en el momento en que se formateo y flasheo la unidad.

Finalmente se procede a instalar el firmware mediante la siguiente línea de comandos:

esptool.py --port COM4 --baud 460800 write_flash --flash_size=detect 0 /Users/cesar/Downloads/esp32-20210902-v1.17.bin

Como se logra apreciar esta línea de comandos también dependerá del usuario, ya que hay al menos 2 parámetros que cambiaran según el puerto de conexión con el micro y la ubicación del Firmware descargado.

```
C:\Windows\system32>esptool.py --port COM4 --baud 460800 write_flash --flash_size=detect 0 /Users/ce
sar/Downloads/esp32-20210902-v1.17.bin
esptool.py v3.1
Serial port COM4
Connecting.....
Detecting chip type... ESP32
Chip is ESP32-D0WDQ6 (revision 1)
Features: WiFi, BT, Dual Core, 240MHz, VRef calibration in efuse, Coding Scheme None
WARNING: Detected crystal freq 41.01MHz is quite different to normalized freq 40MHz. Unsupported cry
stal in use?
Crystal is 40MHz
MAC: 4c:11:ae:7b:bc:0c
Uploading stub...
Running stub...
Stub running...
Changing baud rate to 460800
Changed.
Configuring flash size...
Auto-detected Flash size: 4MB
Flash will be erased from 0x00000000 to 0x00174fff...
Compressed 1527504 bytes to 987584...
Wrote 1527504 bytes (987584 compressed) at 0x00000000 in 26.7 seconds (effective 458.4 kbit/s)...
Hash of data verified.

Leaving...
Hard resetting via RTS pin...
```

Ilustración 31: CMD ejecutando la linea de comando esptool.py --port COM4 --baud 460800 write_flash --flash_size=detect 0 /Users/cesar/Downloads/esp32-20210902-v1.17.bin

Instalación *Jupyter Notebook*: De igual manera todo se realiza a partir de la ventana de comandos, primero se realiza la instalación de *Jupyter,* aunque en algunos casos es necesario actualizar la función pip con el siguiente comando:

python.exe -m pip install --upgrade pip

La línea de comando anterior solo se utilizar en caso de que se solicite automáticamente al intentar ejecutar la siguiente línea de comando. Luego se instala Jupyter.

python.exe -m pip install jupyter

```
C:\Windows\system32>python.exe -m pip install jupyter
Collecting jupyter
  Downloading jupyter-1.0.0-py2.py3-none-any.whl (2.7 kB)
Collecting notebook
  Downloading notebook-6.4.4-py3-none-any.whl (9.9 MB)
     |████████████████████████████████| 9.9 MB 2.2 MB/s
Collecting qtconsole
  Downloading qtconsole-5.1.1-py3-none-any.whl (119 kB)
     |████████████████████████████████| 119 kB 3.3 MB/s
Collecting ipywidgets
  Downloading ipywidgets-7.6.5-py2.py3-none-any.whl (121 kB)
```

Ilustración 32: CMD ejecutando la línea de comando python.exe -m pip install --upgrade pip

Al terminar la descarga e instalación se realiza la actualización del cliente

pip3 install --upgrade jupyter_client

```
C:\Windows\system32>
C:\Windows\system32>pip3 install --upgrade jupyter_client
Requirement already satisfied: jupyter_client in c:\users\cesar\appdata\local\prog
39\lib\site-packages (7.0.5)
Requirement already satisfied: traitlets in c:\users\cesar\appdata\local\programs\
b\site-packages (from jupyter_client) (5.1.0)
Requirement already satisfied: python-dateutil>=2.1 in c:\users\cesar\appdata\loca
python39\lib\site-packages (from jupyter_client) (2.8.2)
Requirement already satisfied: entrypoints in c:\users\cesar\appdata\local\program
lib\site-packages (from jupyter_client) (0.3)
```

Ilustración 33: CMD ejecutando pip3 install --upgrade jupyter_client

Posteriormente se instala *Jupyter Notebook*

pip install jupyter_micropython_kernel

```
C:\Windows\system32>pip install jupyter_micropython_kernel
Collecting jupyter_micropython_kernel
  Downloading jupyter_micropython_kernel-0.1.3.2-py3-none-any.whl (20 kB)
Requirement already satisfied: pyserial>=3.4 in c:\users\cesar\appdata\local\programs\python\python3
9\lib\site-packages (from jupyter_micropython_kernel) (3.5)
Collecting websocket-client>=0.44
  Downloading websocket_client-1.2.1-py2.py3-none-any.whl (52 kB)
     |████████████████████████████████| 52 kB 1.9 MB/s
Installing collected packages: websocket-client, jupyter-micropython-kernel
Successfully installed jupyter-micropython-kernel-0.1.3.2 websocket-client-1.2.1

C:\Windows\system32>python -m jupyter_micropython_kernel.install
Installing IPython kernel spec of micropython
C:\Users\cesar\AppData\Local\Programs\Python\Python39\lib\site-packages\jupyter_micropython_kernel\i
nstall.py:29: DeprecationWarning: replace is ignored. Installing a kernelspec always replaces an exi
sting installation
  k.install_kernel_spec(td, 'Micropython', user=user, replace=True, prefix=prefix)
...into C:\Users\cesar\AppData\Roaming\jupyter\kernels\micropython

C:\Windows\system32>jupyter kernelspec list
Available kernels:
  micropython    C:\Users\cesar\AppData\Roaming\jupyter\kernels\micropython
  python3        C:\Users\cesar\AppData\Local\Programs\Python\Python39\share\jupyter\kernels\python3
```

Ilustración 34: CMD ejecutando pip install jupyter_micropython_kernel

ANEXO 2: GUIA DE INSTALACIÓN MICROPYTHON

Instalación Micropython: Esta es una extensión de Jupyter Notebook que es la que prácticamente nos habilitara el poder programar sobre el microcontrolador.

pip install jupyter_micropython_kernel

```
C:\WINDOWS\system32>
C:\WINDOWS\system32>pip install jupyter_micropython_kernel
Requirement already satisfied: jupyter_micropython_kernel in c:\users\cesar\appdata\local\programs\python\python310\lib\
site-packages (0.1.3.2)
Requirement already satisfied: pyserial>=3.4 in c:\users\cesar\appdata\local\programs\python\python310\lib\site-packages
 (from jupyter_micropython_kernel) (3.5)
Requirement already satisfied: websocket-client>=0.44 in c:\users\cesar\appdata\local\programs\python\python310\lib\site
-packages (from jupyter_micropython_kernel) (1.2.1)
```

Ilustración 35: CMD ejecutando pip install jupyter_micropython_kernel

En este caso debido a que ya se había instalado no se alcanza a ver todo el proceso en la imagen anterior, sin embargo, se agrega como prueba que el comando funciona. Una vez completados todos los pasos anteriores ya se puede utilizar *Jupyter Notebook* con *Micropython* y programar el microcontrolador con la computadora a la cual se está conectado.

Funcionamiento de Jupyter Notebook: Para utilizar Jupyter Notebook basta con escribir su nombre en la línea de comando, y este se abrirá de forma predeterminada utilizando el puerto 8888.

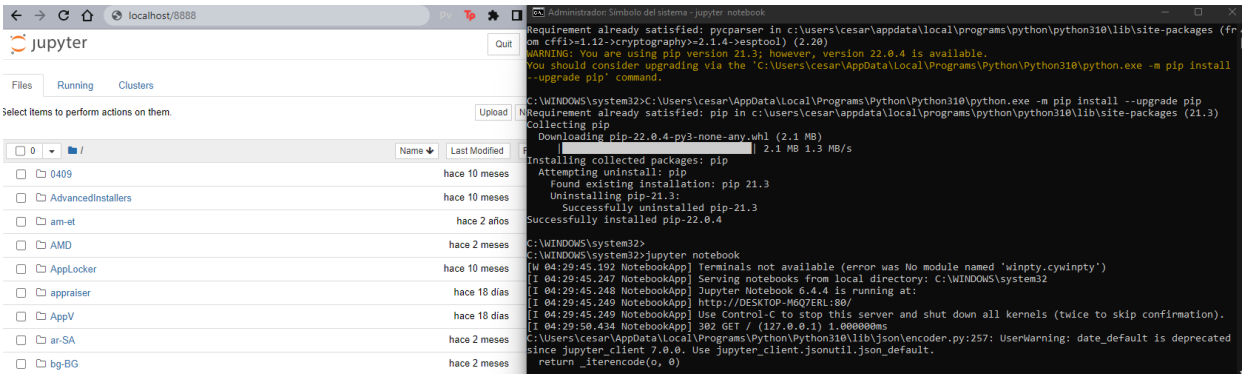

Ilustración 36: Comando de ejecución para Jupyter Notebook y ventana principal del mismo

Para abrir el Notebook y utilizar *Micropython* basta con pulsar la opción que dice “New” dentro de la ventana principal de *Jupyter Notebook* y a continuación seleccionar la opción de “MicroPython – USB”, a continuación, se abrirá el Notebook en donde se podrán escribir las rutinas para poder controlar el equipo mediante internet.

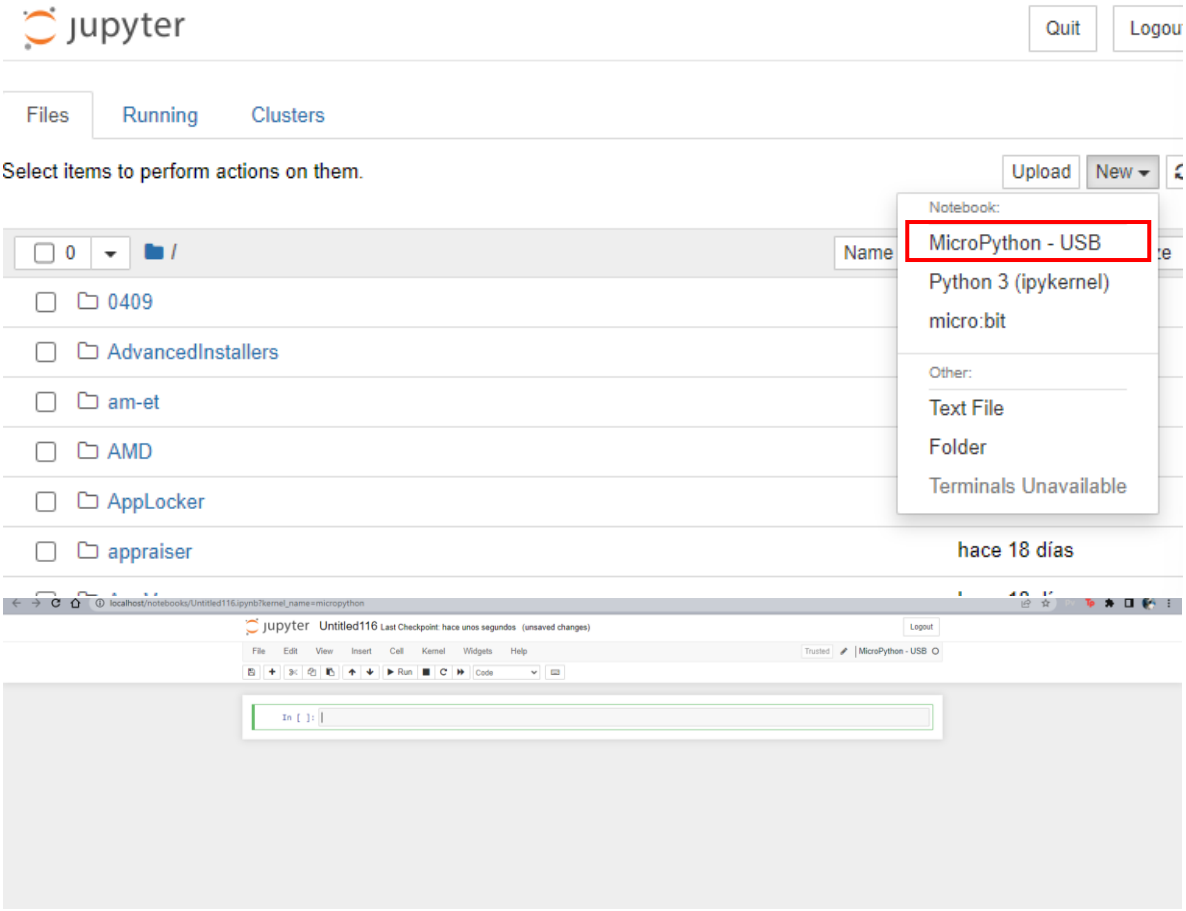

Ilustración 37: MicroPython-USB y Notebook generado.

Modificaciones en la configuración por defecto de *Jupyter Notebook:* Sera necesario realizar algunas modificaciones al archivo de configuración de *Jupyter Notebook* para permitirle a este ser visto desde otros equipos de la misma red, para esto nos dirigimos a la carpeta en donde se instaló *Jupyter Notebook,* la cual si no se cambió debería encontrarse en la siguiente dirección: C:\Users\usuario\.jupyter dentro se encontrara un archivo con el siguiente nombre: jupyter_notebook_config.py, el cual podremos editar con el bloc de notas o cualquier otro editor de código.

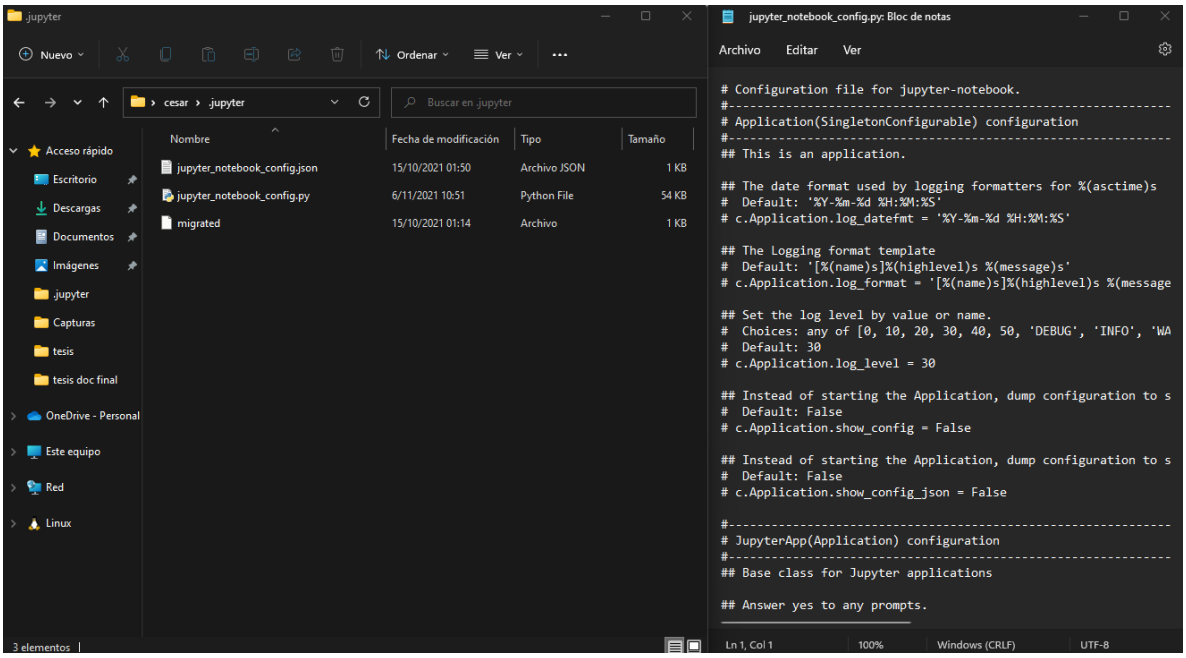

Ilustración 38: Archivo de configuración de Jupyter Notebook

Dentro de este archivo de configuración será necesario realizar los siguientes cambios:

1- Buscar la siguiente instrucción #c.NotebookApp.allow_origin = '*' y eliminar el símbolo de “#”, esto permitirá que Jupyter Notebook permita recibir solicitudes de todos los orígenes.

```
## Set the Access-Control-Allow-Origin header
#
#         Use '*' to allow any origin to access your server.
#
#         Takes precedence over allow_origin_pat.
#  Default: ''
c.NotebookApp.allow_origin = '*'
```

Ilustración 39: Instrucción modificada en el archivo de configuración

2- Buscar la siguiente instrucción c.NotebookApp.ip = '0.0.0.0' y eliminar el símbolo de “#”, esto permitirá que Jupyter Notebook pueda ser accedido desde cualquier IP que tenga acceso a este servicio.

```
## The IP address the notebook server will listen on.
#  Default: 'localhost'
c.NotebookApp.ip = '0.0.0.0'
```

Ilustración 40: Instrucción modificada en el archivo de configuración

3- Configuramos el puerto por el cual se proveerá el servicio de *Jupyter Notebook,* en este caso utilizaremos el puerto 80 ya que de esta manera solo es necesario escribir la IP en el buscador, y ya no será necesario verificar el puerto que se está utilizando.

```
## The port the notebook server will listen on (env: JUPYTER_PO
#  Default: 8888
c.NotebookApp.port = 80
```

Ilustración 41: Selección de puerto dentro del archivo de configuración.

De esta manera ya queda completamente configurado *Jupyter Notebook,* y ya puede ser utilizado desde cualquier dispositivo dentro de la misma red LAN, posteriormente se explicará la instalación de Google Cloud para poder utilizar este servicio desde cualquier red.

ANEXO 3: GUIA DE CREACIÓN DE SERVICIO EN GOOGLE CLOUD Y OPEN VPN

Para utilizar este servicio de Google necesitaremos crear una cuenta Gmail, la cual también se utilizará para el almacenamiento de datos en Google Drive, para este trabajo de graduación se utiliza la siguiente cuenta:

user: mesamovil.ues2022@gmail.com

pass: tesis2022

Con esta cuenta se registra en Google Cloud y se empieza a configurar para poder utilizar la instancia virtual como servidor.

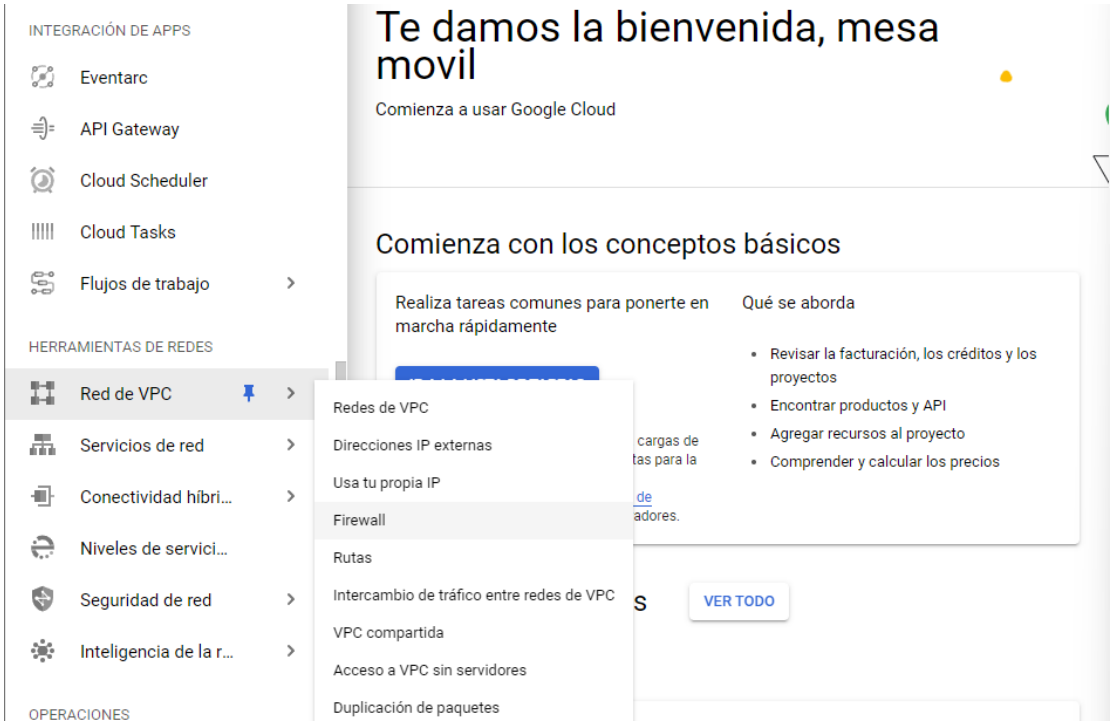

Ilustración 42: Red de VPC y Firewall

Primero se crea una regla de firewall para que podamos acceder sin problema a la IP externa que nos asigne Google Cloud, esto se encuentra en el menú en el apartado de Herramientas de redes en la opción de Red de VPC.

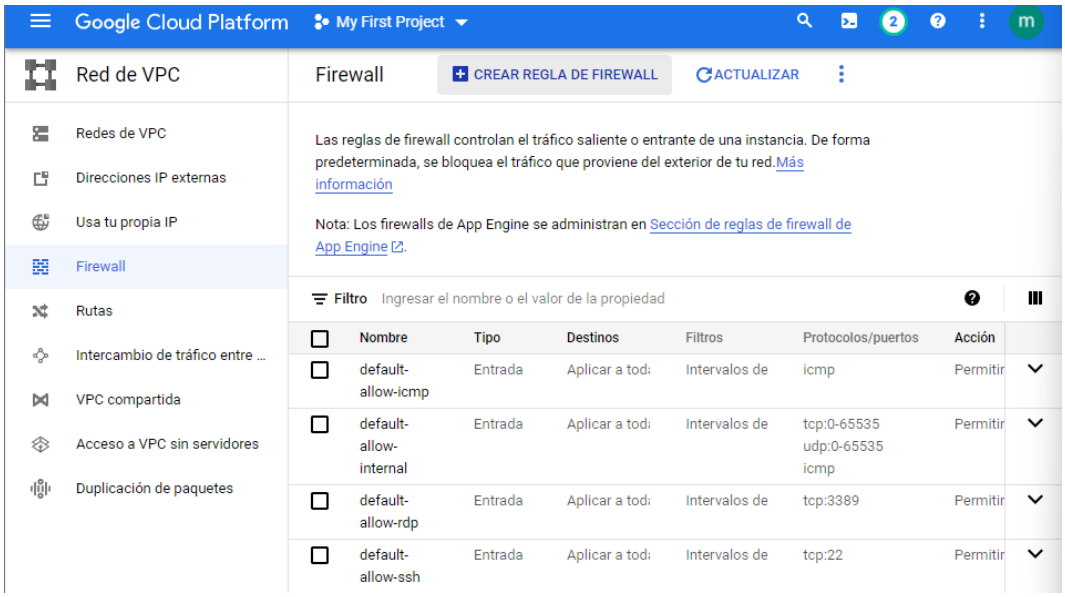

Ilustración 43:Reglas de firewall por defecto

Primero se utiliza la opción de crear una nueva regla de de firewall y posteriormente se configura como aparece en las siguientes imágenes.

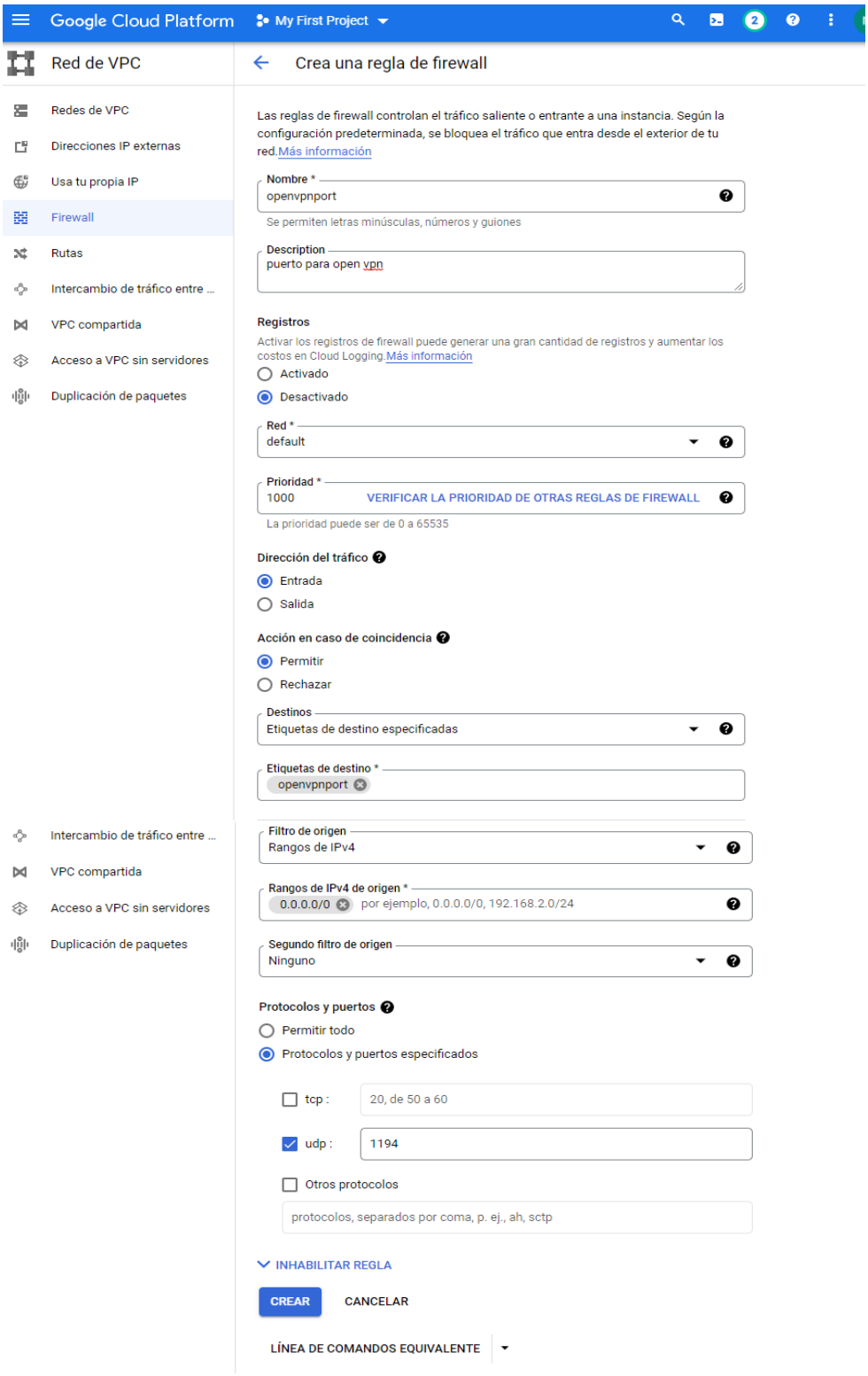

Ilustración 44: Configuración de nuevo Firewall para permitir solicitudes de IP externas.

Al final debería aparecer una nueva regla de firewall en el menú inicial.

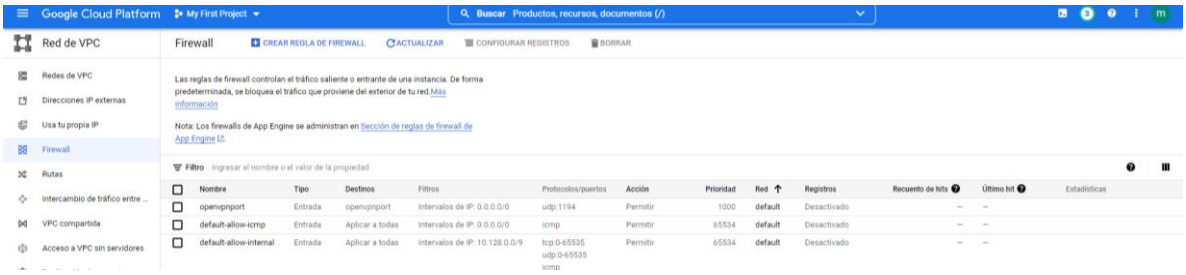

Ilustración 45: Nueva regla de Firewall "openvpnport"

Una vez creada esta regla del firewall ya se pueden crear las instancias de *virtual machine,* la cual nos servirá de servidor para podernos conectar desde distintas redes mediante una VPN. Para esto nos dirigimos al menú principal, en el apartado de "Compute Engine" y seleccionamos la opción de "Instancias de VM".

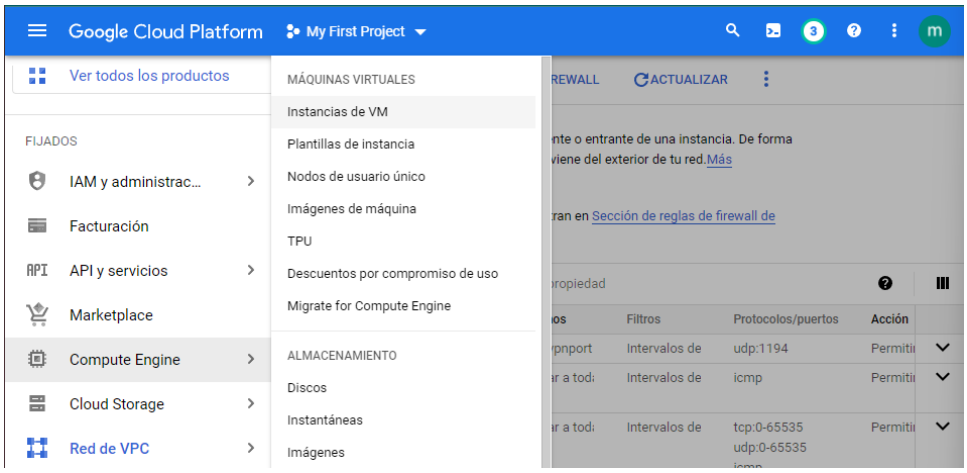

Ilustración 46: Menú para creación de Instancia de VM

En el menú de configuración se selecciona "Ubuntu" como disco de arranque, ya que este es gratuito y cumple con los requisitos necesarios para ser utilizado como servidor.

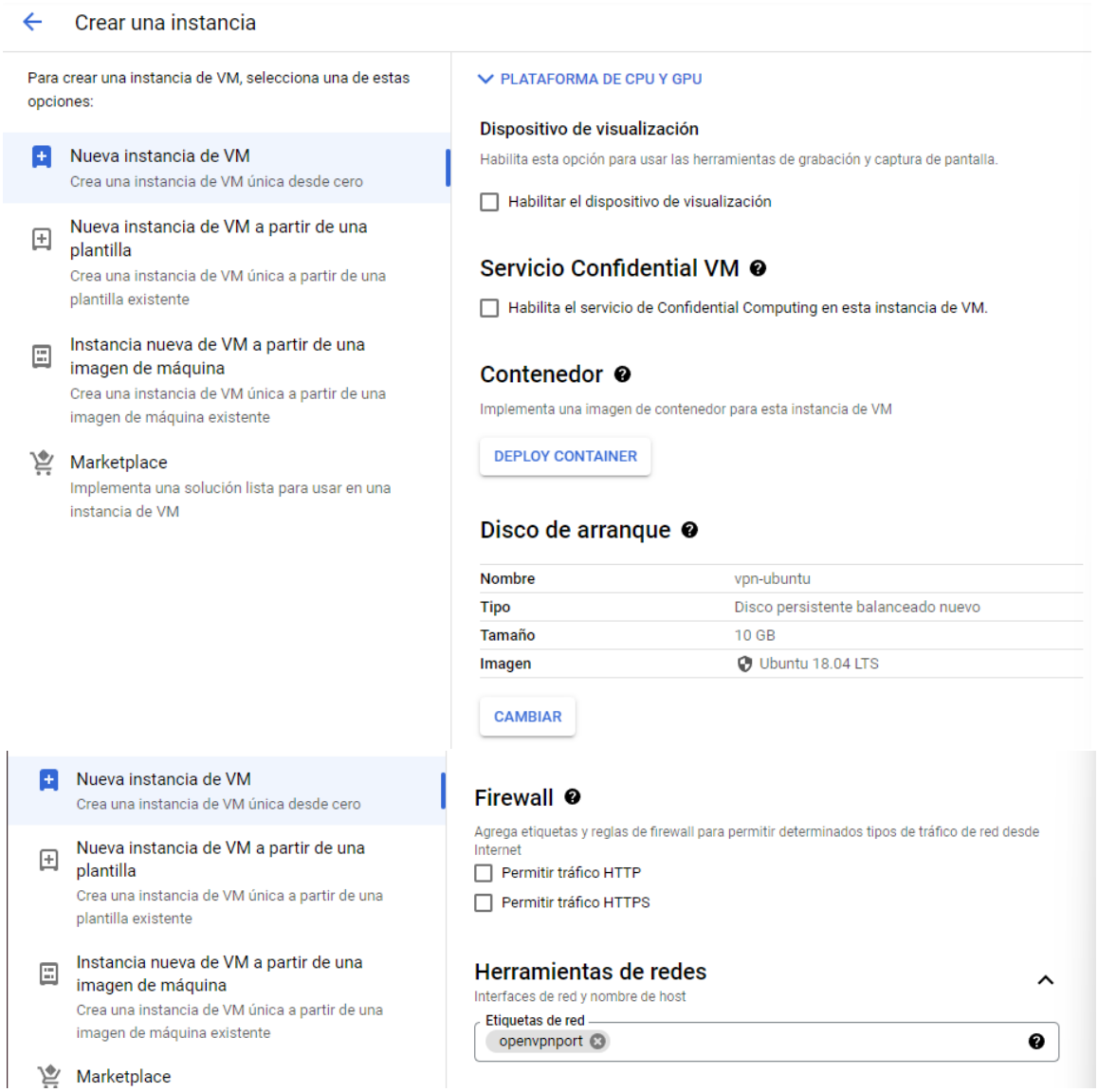

Ilustración 47: Configuración de Virtual Machine con Ubuntu

Luego de guardar se inicia la consola al darle click en “conectar”, al iniciar la consola se deben ejecutar los siguientes comandos para actualizarla:

sudo apt-get update

sudo apt-get upgrade

Y finalmente se instala OpenVPN, el cual nos permitira establecer la conexión a través de la instancia virtual de Google Cloud.

sudo wget https://git.io/vpn -O openvpn-install.sh && sudo bash openvpn-install.sh

Este comando podría cambiar, lo importante es encontrar una web donde se descargue el archivo.

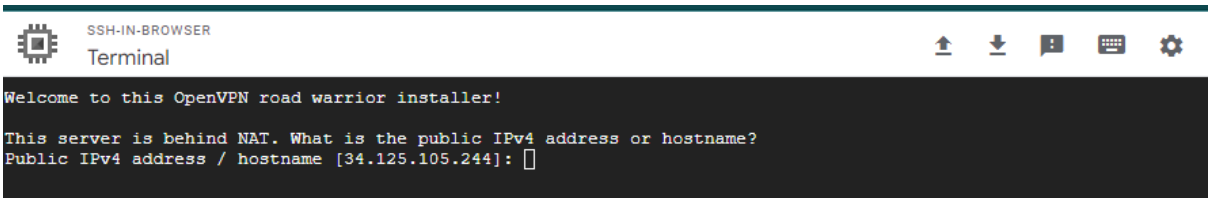

Ilustración 48: CMD ejecutando la línea de comando de instalación de OpenVPN

Se configura la ip externa que nos asigna la instancia virtual, la cual podemos observar en el menú de instancias, luego se selecciona protocolo UDP, utilizamos también el puerto recomendado y escogemos el servidor DNS, en este caso se utilizó el de Google.

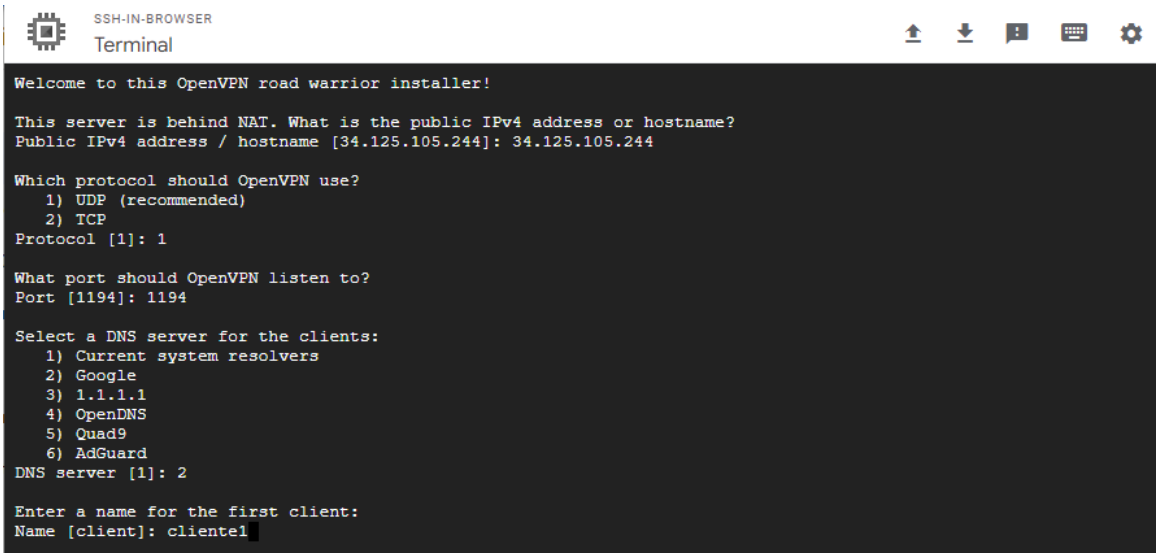

Ilustración 49: Configuración de instalación de OpenVPN

Como último paso se agregan los usuarios utilizando la siguiente línea de comando:

sudo bash openvpn-install.sh

Estos usuarios serán importados al programa OpenVPN instalado en cada equipo que desee conectarse a la red. Para eso se descargan mediante la opción de descargar que aparece en la parte superior izquierda.

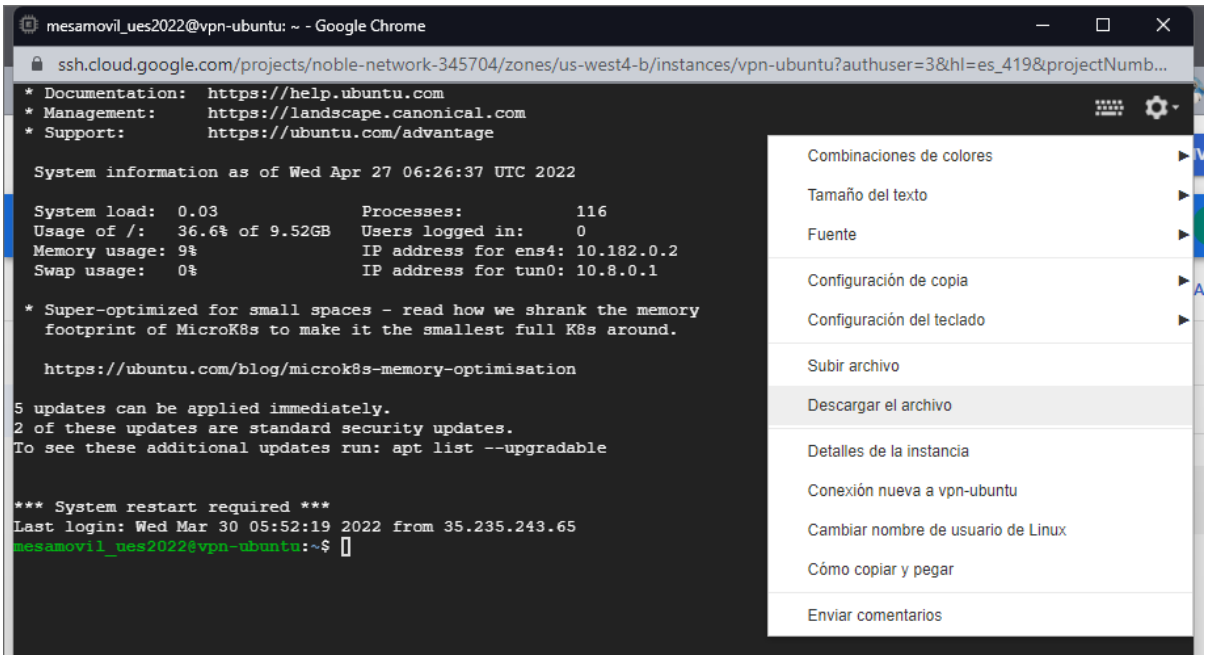

Ilustración 50: Menú para descarga de archivo

Una vez seleccionada esa opción de descarga será necesario escribir el nombre y la ubicación del archivo. Con esto ya estaremos listos para utilizar Open VPN y lanzar el servicio de Jupyter Notebook.

ANEXO 4: CONFIGURACIÓN DE IFTTT

Este servicio es clave para el almacenamiento de los datos dentro de Google Drive, como primer paso accederemos a su sitio web:

https://ifttt.com/explore

Una vez dentro nos moveremos a la sección de *My applets* y crearemos un applet nuevo. A continuación, seleccionaremos los servicios que se muestran en la ilustración 51 y procederemos a editarlas.

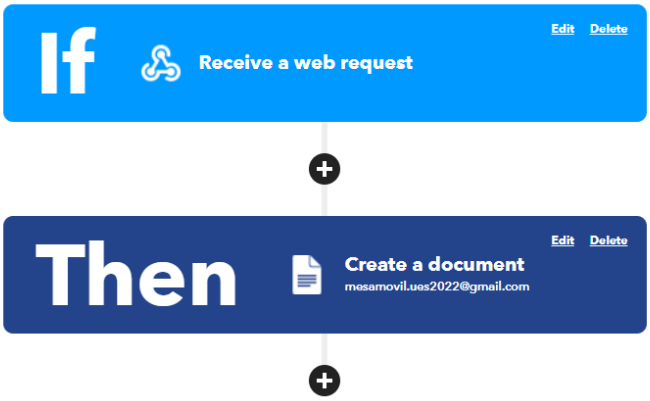

Ilustración 51: Configuración de Applet en IFTTT

Para la configuración del *if* solo será necesario la configuración de un nombre para el evento. Sin embargo, para la configuración de *Then* es donde se define el correo electrónico que se utilizara para Google Drive, el número de variables que se guardaran y otros detalles como la carpeta, nombre del evento, etc. Esta configuración es mostrada en la ilustración 52.

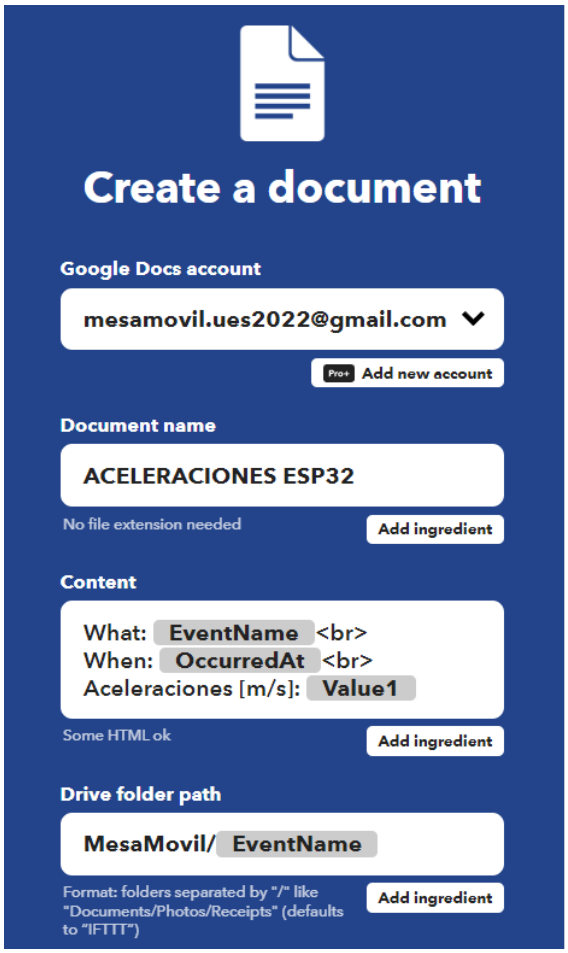

Ilustración 52: Configuración de Then en IFTTT

Como ultimo paso utilizaremos el siguiente link para acceder a la documentación del servicio.

https://ifttt.com/maker_webhooks

Una vez dentro de la sección de documentación podremos acceder a la contraseña del servicio, la cual nos ayudara a poder enviar la solicitud de accionamiento del servicio mediante el código en Jupyter Notebook. La clave para este servicio en especifico es la que se muestra en la ilustración 53.

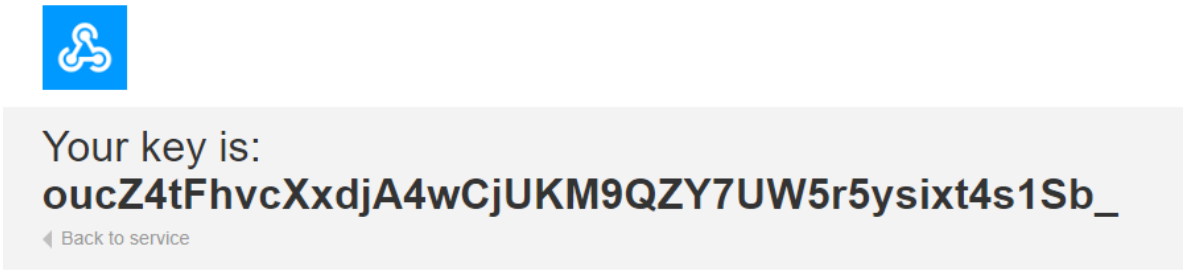

Ilustración 53: Clave IFTTT del Applet configurado para este trabajo de graduación

Esta clave se utiliza en el código de la sección 3.4.5, con esto ya quedara habilitado la comunicación entre Google Drive y Jupyter Notebook.