Escuela Superior
de Ingeniería y Tecnología
Universidad de La Laguna

Dron Autónomo de Reconocimiento de Personas

Memoria de la asignatura de Trabajo de Fin de Grado del Grado en Ingeniería Informática

Memoria de Trabajo de Fin de Grado

Autor: Daniel Arbelo Hernández

Gmail: alu0101117621@ull.edu.es

La presente memoria ha sido realizada por el estudiante Daniel Arbelo Hernández para ser presentada como Memoria de Trabajo de Fin de Grado en el Grado en Ingeniería Informática por la Universidad de La Laguna.

El Trabajo de Fin de Grado ha sido realizado sobre el reconocimiento de imágenes por dron y han sido tutorizadas por D. Cándido Caballero-Gil, en calidad de tutor académico y por D. Ricardo Aguasca Colomo en calidad de cotutor académico.

Agradecimientos

Quiero expresar mi más sincero agradecimiento a mis tutores por su apoyo durante todo el proceso de mi trabajo de fin de grado. A pesar de los ajustados plazos de entrega y los desafíos de última hora, se adaptaron y me brindaron su guía y supervisión para lograr el éxito.

Quiero reconocer especialmente a D. Javier Correa de Acudropol, empresa dedicada al montaje de drones para la policía, a quien la Universidad de La Laguna le encargó el montaje de su dron para investigación y trabajos de fin de grado, por su generosa colaboración. Su ayuda, ideas y disposición para compartir el material necesario fueron fundamentales para facilitar el desarrollo de mi trabajo. La disponibilidad de la controladora del dron, el router, el GPS y las baterías, así como el cable que hizo especialmente para la transmisión entre la controladora del dron y la Jetson Nano, y su amplio conocimiento sobre drones, calibración, conexiones, entre otros aspectos, fueron elementos esenciales que contribuyeron en gran medida a mi investigación.

Índice de contenido

1. Introducción

El presente trabajo de fin de grado se centra en el desarrollo de un sistema de reconocimiento de imágenes utilizando drones para identificar y rastrear personas. El objetivo principal es permitir al dron detectar a una persona en determinada área y, una vez reconocida, capturar una fotografía de alta resolución y almacenarla junto con su ubicación geográfica precisa.

El sistema está diseñado para proporcionar un enfoque eficiente y preciso en la vigilancia y monitoreo de áreas extensas, brindando apoyo en diversas situaciones, como búsqueda y rescate, seguridad y seguimiento de individuos en entornos remotos o de difícil acceso.

Cuando el dron detecta a una persona, inicia automáticamente la secuencia de captura de imágenes. Además de la primera fotografía, el dron continuará volando en círculos alrededor de la persona identificada, capturando imágenes adicionales desde diferentes ángulos y guardando la ubicación correspondiente. Esto proporciona una visión más completa de la persona en cuestión y permite una mejor comprensión de su entorno.

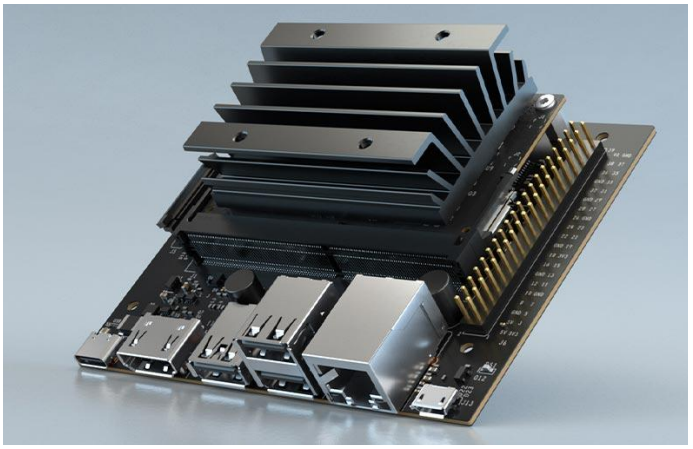
Es importante destacar que el operario de tierra desempeña un papel crucial en este sistema, ya que es quien describe la misión que realizará el dron, incluyendo la ruta que seguirá. Una vez que el dron ha obtenido suficientes imágenes, o si el operario lo decide, puede optar por continuar con la misión, previamente descrita, o permitir que el dron siga orbitando alrededor de la persona objetivo. Alternativamente, el operario también tiene la opción de interrumpir la misión, tomando la decisión de que el dron regrese al punto de partida y aterrice de manera segura.

Este trabajo de fin de grado integra tecnologías de reconocimiento de imágenes, control de vuelo de drones, geolocalización y comunicación entre el dron y el operario de tierra. Se espera que el sistema propuesto brinde una solución efectiva y versátil para el monitoreo y seguimiento de personas en diversas aplicaciones, con el objetivo de mejorar la eficiencia y seguridad en estas tareas.

2. Materiales utilizados

2.1. Jetson Nano

NVIDIA® Jetson Nano™ es una computadora pequeña y potente diseñada para aplicaciones integradas de inteligencia artificial. Debido a su potencia y tamaño compacto, resulta ideal para este proyecto, ya que el dron tiene restricciones de peso y otros componentes que también deben ser considerados. En este proyecto, se ha utilizado el modelo de 2GB, el cual presenta algunas diferencias respecto a sus contrapartes de 4 GB, como la cantidad de puertos disponibles. A pesar de tener un poco menos de potencia, se ajusta perfectamente a los requisitos de este proyecto.



Nvidia Jetson Nano 2GB Developer Kit

2.2. Pixhawk

Es la controladora que Acudropol suministró para realizar pruebas y desarrollar el proyecto antes de llevarlo a cabo en la realidad. Esta controladora resulta más económica en comparación con la CUAV V5+, la cual tiene un costo aproximado de 600 euros.

La controladora Pixhawk es una plataforma de control de vuelo ampliamente utilizada en sistemas de vehículos aéreos no tripulados (UAV) o drones. Diseñada para ofrecer un rendimiento y una fiabilidad excepcionales, la controladora Pixhawk es conocida por su versatilidad y su capacidad para manejar una amplia variedad de aplicaciones y configuraciones.

La controladora Pixhawk utiliza un conjunto de sensores, incluyendo acelerómetros, giroscopios y magnetómetros, para medir y controlar la actitud y posición del dron en vuelo. También cuenta con puertos de entrada/salida (I/O) para conectar y comunicarse con diversos componentes y periféricos, como receptores GPS, módulos de telemetría, sensores adicionales, cámaras, entre otros.

Pixhawk 2.4.8

2.3. Router tp-link

Es el router que se utiliza para conectarlo a la Jetson nano y que desde un ordenador (estación de tierra) se puede conectar a esta, que hace de puente entre la pixhawk y el router, redirigiendo los paquetes uart.

3. Construcción del dron

3.1. Montaje

En cuanto al punto de construcción del dron, se ha delegado esta tarea a ACUDROPOL, quienes poseen amplios conocimientos en este campo y cuentan con experiencia en la selección de materiales adecuados. La Universidad de La Laguna adquirió un dron previamente y le encomendó a Javier Correa su construcción, considerando que implica el uso de materiales costosos y requiere un nivel de experiencia especializada.

3.2. Instalación de software en la Controladora

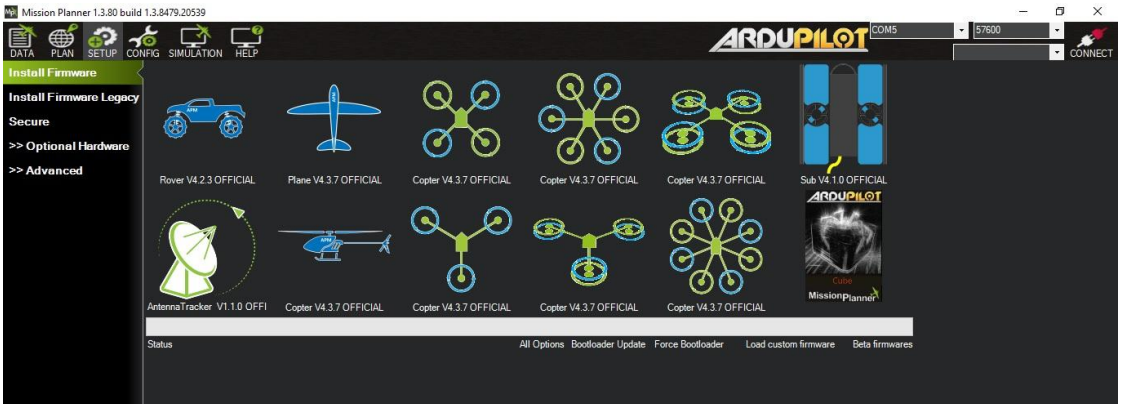
Para la instalación del software en la controladora del dron utilizando Mission Planner. Brevemente, el proceso implica seguir una serie de pasos para cargar y configurar el firmware en la controladora.

Para instalar el firmware en la controladora, es necesario seguir algunos pasos clave. En primer lugar, instalar el software de Mission Planner en el ordenador. El enlace de descarga se encuentra en el punto 21 de la bibliografía o en el sitio web oficial de Mission Planner.

Una vez que tienes el software de Mission Planner instalado, es necesario tener a mano la controladora y un cable microUSB de transferencia de datos. Este cable se utilizará para conectar la controladora al ordenador y permitir que el programa Mission Planner la reconozca correctamente.

Una vez conectada se va a SETUP y dentro de aquí a Install Firmware, todo esto es importante hacerlo sin darle a conectar a la controladora, arriba a la derecha de la imágen, ya que no funcionará. Después de esto se escoge Copter V4.3.7 OFFICIAL, ya que es el firmware para un Cuadricóptero, como el que se utiliza en este proyecto. Una vez instalado, se reinicia la controladora, y ahora solo quedaría calibrar, y hacer la configuración necesaria para que funcione, pero como no se dispone ni de

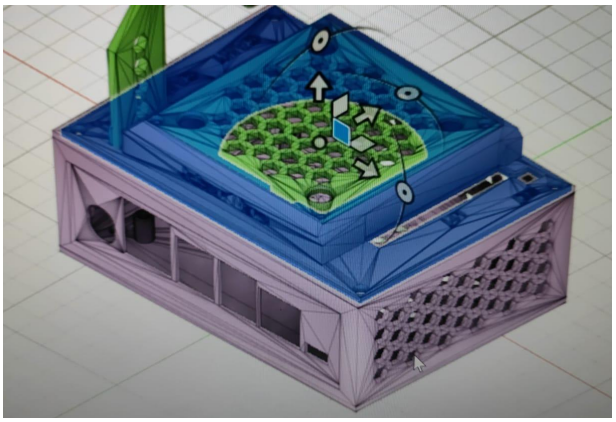
mandos ni el dron, pues esta parte de calibración se delega a ACUDROPOL.



Mission Planner instalación de Firmware

3.3. Soporte de Jetson Nano en el Dron

Además, en colaboración con ACUDROPOL, se ha trabajado en la adaptación de un soporte impreso en 3D para acoplar la Jetson Nano al dron. Se proporcionó un modelo encontrado en una página especializada, pero se requerían modificaciones ya que estaba diseñado para la Jetson Nano de 4GB. ACUDROPOL ha asumido la responsabilidad de realizar las adaptaciones necesarias para asegurar una integración correcta entre la Jetson Nano y el dron.



Soporte 3D para la Jetson Nano (Bibliografía pto.2)

3.4. Interferencias

Durante la investigación, se ha encontrado que algunos usuarios han experimentado interferencias entre la Jetson Nano y la controladora o el GPS del dron en casos específicos. Para evitar problemas y proteger los costosos componentes, los usuarios han aplicado una solución simple: envolver la Jetson Nano con papel de aluminio, creando una especie de jaula de Faraday. Esto protege contra los campos eléctricos estáticos y evita posibles interferencias electromagnéticas. Esta precaución adicional proporciona estabilidad en la comunicación entre los componentes del dron. (Bibliografía pto.1)

4. Programas de Control de Vuelo

A lo largo del proyecto, se han explorado diferentes programas para conectar a la controladora Pixhawk desde la estación de tierra y realizar diversas tareas. Después de evaluar varias opciones, finalmente se optó por utilizar Mission Planner para instalar el firmware en la controladora. En el punto 3.2 se adjuntan las instrucciones detalladas sobre cómo llevar a cabo este proceso.

Inicialmente, se intentó utilizar QGroundControl para instalar el firmware en la controladora. Sin embargo, se encontraron dificultades al intentar utilizar el puerto de telemetría 2 para conectarla a la Jetson Nano. Solo se pudo utilizar el puerto de telemetría 1, lo que limitaba la capacidad de comunicación. Debido a estos problemas, se decidió utilizar el firmware de Mission Planner, ya que permite establecer una conexión estable y acceder a todas las funcionalidades necesarias. Además, tras investigar otros proyectos similares, se constató que muchos usuarios también han optado por utilizar Mission Planner para este proceso.

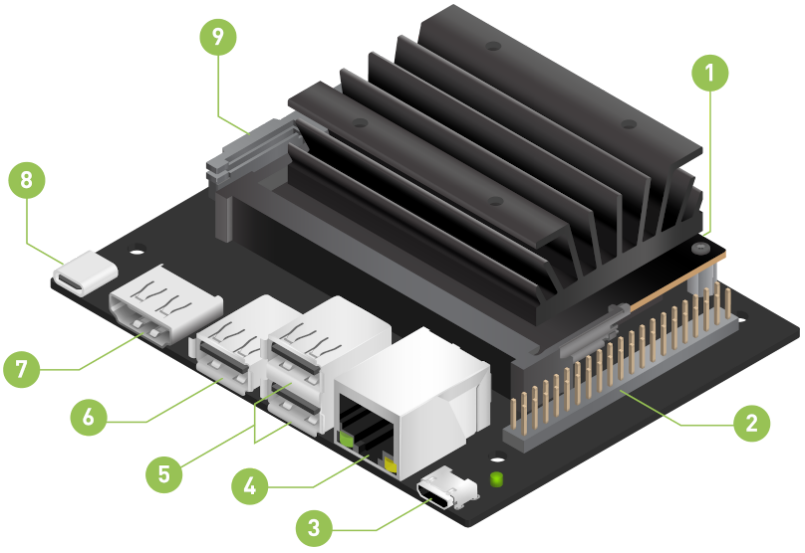
Por otro lado, en cuanto a la planificación y ejecución de misiones, se ha utilizado QGroundControl. En el Anexo 2 se incluyen instrucciones detalladas sobre cómo crear y llevar a cabo misiones utilizando esta plataforma, la cual se consideró más intuitiva para este proceso. Además, dado que el operario del dron estaba más familiarizado con esta plataforma, optamos por utilizarla en conjunto para garantizar una mejor

coordinación y eficiencia en las tareas de planificación y ejecución de misiones.

5. Jetson Nano

5.1. Instalación de Sistema Operativo

La instalación del sistema operativo en la Jetson Nano es esencial para aprovechar su potencial. Para comenzar, hay que descargar la imagen del sistema operativo recomendada desde el sitio web oficial de NVIDIA. Luego, utilizar un programa de escritura de imágenes para grabar la imagen en una tarjeta microSD. Insertar la tarjeta en la Jetson Nano, encenderla y el sistema operativo se cargará. En el apartado 3 de la bibliografía, se proporciona un enlace donde se detalla paso a paso cómo realizar la instalación del sistema operativo en la Jetson Nano para el modelo utilizado en el proyecto.



Nvidia Jetson Nano 2GB Developer Kit

La configuración de la Jetson Nano requiere una tarjeta SSD en el punto 1, una fuente de alimentación conectada al punto 8, un teclado y ratón conectados a los puertos USB, y un monitor conectado al puerto

HDMI en el punto 7. Estos elementos son esenciales para asegurar el funcionamiento adecuado y la interacción con el sistema.

5.2. Conectarse por SSH

Una vez que el sistema operativo ha sido configurado en la Jetson Nano, es posible establecer una conexión mediante un cable micro USB de transferencia de datos. Este cable se conecta al punto 3 de la placa y al ordenador.

Para configurar la Jetson Nano de manera más sencilla, se puede utilizar SSH. Se puede establecer una conexión SSH siguiendo el siguiente formato: ssh '<username>@192.168.55.1', donde '<username>' debe ser reemplazado por el nombre de usuario configurado en la Jetson Nano. Esta conexión permite acceder a la placa y realizar configuraciones y ajustes necesarios. En este proyecto en particular, se ha utilizado Visual Studio Code para establecer la conexión SSH.

Además, si la Jetson Nano está conectada a la misma red, es posible acceder a ella utilizando la dirección IP asignada en la red. Esto permite conectar y configurar la placa desde cualquier otro dispositivo en la misma red.

En el Anexo 1 se explica cómo realizar la conexión con Visual Studio Code.

5.3. Configurando Jetson Nano

Antes de continuar con el proyecto, es importante realizar algunas instalaciones y configuraciones adicionales para evitar posibles fallos y garantizar la compatibilidad entre las diferentes versiones de software. A continuación, se detallan los pasos necesarios:

1. Actualizar los paquetes del sistema operativo ejecutando el siguiente comando en la terminal:

```
Unset
sudo apt-get update
```

Esto asegurará que todos los paquetes estén actualizados a sus últimas versiones.

2. Instalar Git, en el Anexo 3 se muestra como hacerlo.

3. Instalar algunas bibliotecas y herramientas adicionales necesarias para el proyecto. Se deben ejecutar los siguientes comandos uno por uno en la terminal:

```
Unset
sudo pip install future
sudo pip install pyserial
sudo pip install MAVProxy
```

Estas instalaciones proporcionarán funcionalidades adicionales y la compatibilidad necesaria para el desarrollo y la comunicación entre la Jetson Nano y la controladora Pixhawk.

4. Instalar Python 3.7 en la Jetson Nano debido a algunas necesidades específicas del proyecto. La Jetson Nano viene con Python 2 y Python 3.6 preinstalados. Sin embargo, Python 2 está desactualizado y ya no recibe soporte y Python 3.6 no es compatible con algunas bibliotecas y herramientas utilizadas en el proyecto, como MAVSDK.

```
Unset
sudo apt-get update
sudo apt-get install python3.7
```

También es importante instalar el paquete de desarrollo (dev package) para asegurar que Jetson-Inference reconozca y se instale correctamente en Python 3.7 en el apartado de reconocimiento de personas. Para hacerlo:

```
Unset
sudo apt-get install python3.7-dev
```

Y para comprobar que se ha instalado correctamente, escribir el siguiente comando

```
Unset
dpkg -l | grep python3.7-dev
```

Esto buscará en la lista de paquetes instalados si el paquete python3.7-dev está presente.

Si el paquete está instalado, se obtendrá una línea de salida que indica su versión y detalles de instalación. Por ejemplo, algo como esto:

```
Unset
ii python3.7-dev 3.7.12-1+b3 amd64 header files and
a static library for Python (v3.7)
```

La razón principal para instalar Python 3.7 es que MAVSDK, que se utiliza para interactuar con MAVLink en Python, requiere Python 3.7 o una versión posterior.

Además, Jetson-Inference, que se utiliza para el reconocimiento de personas en el proyecto, también es compatible hasta Python 3.7. Por lo tanto, para asegurar la compatibilidad con todas las herramientas y bibliotecas utilizadas en el proyecto, es necesario utilizar Python 3.7 en lugar de las versiones preinstaladas.

También es posible establecer Python 3.7 como la versión predeterminada para los comandos "python3" y "python" en el sistema utilizando los siguientes comandos:

```
Unset
sudo ln -sf /usr/bin/python3.8 /usr/bin/python
sudo ln -sf /usr/bin/python3.8 /usr/bin/python3
```

Es importante seguir estos pasos para asegurarse de que todas las dependencias estén correctamente instaladas y configuradas, lo que ayudará a evitar problemas de compatibilidad y garantizará un desarrollo fluido del proyecto.

5.4. Modelo de reconocimiento de personas

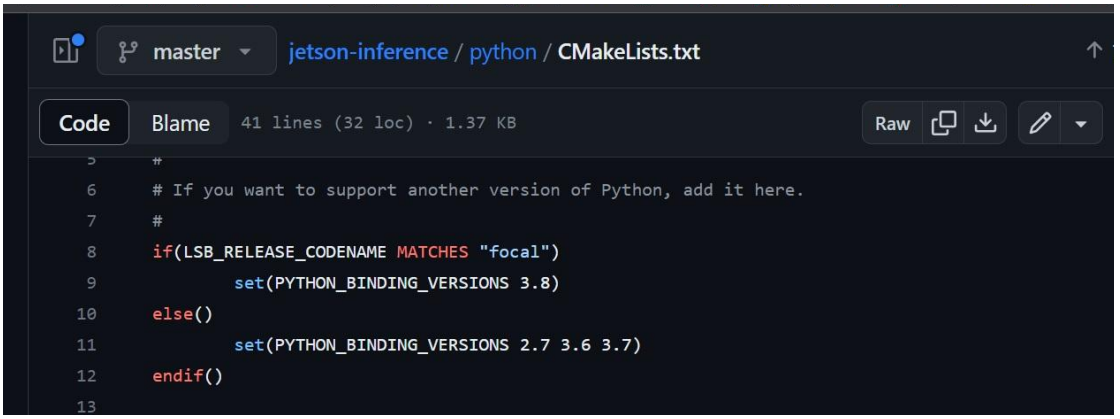
Tras explorar varios modelos de reconocimiento de objetos y personas, se ha descubierto que el proyecto de jetson-inference ofrece opciones destacadas. Entre los diversos proyectos desarrollados por NVIDIA para la Jetson Nano, se encontraron modelos de reconocimiento de personas que ofrecen resultados satisfactorios y no consumen todos los recursos del sistema como otros modelos probados. Esto evita que la Jetson Nano se sobrecargue y que el programa se bloquee durante la ejecución.

En el apartado 5 de la bibliografía se puede encontrar un tutorial detallado sobre el uso de jetson-inference, que proporciona una guía paso a paso para implementar el reconocimiento de objetos. Además, en el punto 4 de la bibliografía se encuentra el repositorio de GitHub relacionado con este proyecto, donde se puede acceder al código fuente, ejemplos y recursos adicionales.

En el repositorio del proyecto, específicamente en la sección de "System Setup", se encuentran tres enlaces que proporcionan diferentes métodos para la instalación y construcción del proyecto. Se puede seleccionar el método que se prefiera, pero en este proyecto, se ha utilizado "Building the project from Source". En el apartado 6 de la bibliografía se encuentra el enlace que detalla paso a paso como hacerlo.

Antes de proceder con esta opción, es importante asegurarse de haber instalado todos los componentes mencionados en el punto 5.3 de la documentación, especialmente Python 3.7 y el paquete de desarrollo (dev package). Esto es necesario para que el proyecto pueda detectar y crear las bibliotecas adecuadas para esta versión específica de Python.

Si exploramos la carpeta "python" dentro del repositorio, en el archivo "CMakeLists.txt", en la línea 11, podremos ver las versiones de Python que son compatibles y admitidas por el proyecto.



Versiones de python que soporta jetson-inference

Si se realiza una modificación en el archivo "CMakeLists.txt" y se agrega una versión específica de Python, como Python 3.8 u otra, se espera que el proyecto se instale y compile correctamente para esa versión. Según la información proporcionada por el propietario del repositorio en algunos foros de discusión, esta modificación debería permitir el uso de la versión deseada de Python.

Sin embargo, es importante tener en cuenta que los resultados pueden variar y no siempre se garantiza un funcionamiento correcto. Aunque el propietario del repositorio haya mencionado esta posibilidad, en ocasiones las configuraciones y dependencias pueden ser complejas y pueden surgir problemas inesperados.

Si en el futuro se desea ampliar este proyecto y explorar la posibilidad de utilizar una versión diferente de Python, puede ser recomendable investigar más a fondo sobre este tema. Se pueden consultar foros de discusión, documentación adicional y recursos relacionados para obtener más información y entender mejor las implicaciones y posibles soluciones. En el punto 7 de la bibliografía se encuentra el enlace a este foro.

5.5. Conectar Jetson Nano a Pixhawk

Para establecer la conexión entre la Jetson Nano y la Pixhawk, se utiliza una conexión UART. Esta conexión se realiza mediante los pines RX (recepción) y TX (transmisión) de la Jetson Nano, los cuales se pueden

identificar en el esquema de la Jetson Nano en el punto 5.1, en el punto 2. Además, es importante conectar un cable a tierra para asegurar una conexión estable.

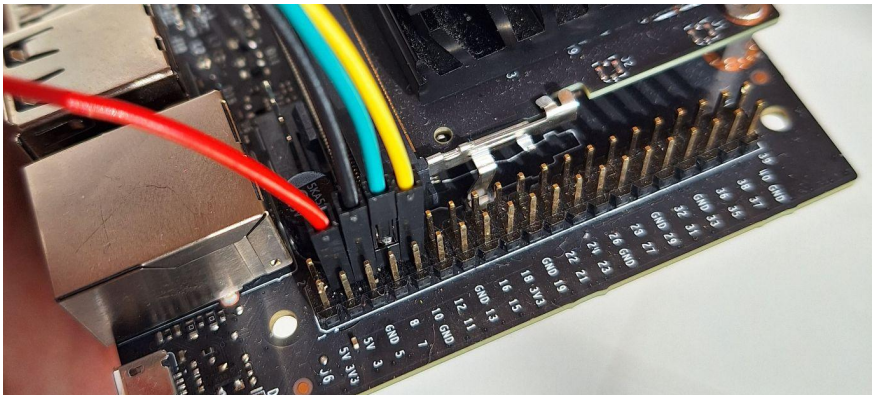
SoC GPIO	Linux GPIO #	Alternate Function	Default Function			Default Function	Alternate Function	Linux GPIO #	SoC GPIO
			3.3 VDC	1	2	5 VDC			
PJ.03	75	GPIO	I2C1_SDA	3	4	5 VDC			
PJ.02	74	GPIO	I2C1_SCL	5	6	GND			
PBB.00	216	AUD_CLK	GPIO	7	8	UART1_TXD	GPIO	48	PG.00
			GND	9	10	UART1_RXD	GPIO	49	PG.01
PG.02	50	UART1_RTS	GPIO	11	12	GPIO	I2S0_SCLK	79	PJ.07
PB.06	14	SPI1_SCK	GPIO	13	14	GND			
PY.02	194		GPIO	15	16	GPIO	SPI1_CS1	232	PDD.00
			3.3 VDC	17	18	GPIO	SPI1_CS0	15	PB.07
PC.00	16	SPI0_MOSI	GPIO	19	20	GND			
PC.01	17	SPI0_MISO	GPIO	21	22	GPIO	SPI1_MISO	13	PB.05
PC.02	18	SPI0_SCK	GPIO	23	24	GPIO	SPI0_CS0	19	PC.03
			GND	25	26	GPIO	SPI0_CS1	20	PC.04
PB.05	13	GPIO	I2C0_SDA	27	28	I2C0_CLK	GPIO	18	PC.02
PS.05	149	CAM_MCLK	GPIO	29	30	GND			
PZ.00	200	CAM_MCLK	GPIO	31	32	GPIO	PWM	168	PV.00
PE.06	38	PWM	GPIO	33	34	GND			
PJ.04	76	I2S0_FS	GPIO	35	36	GPIO	UART1_CTS	51	PG.03
PB.04	12	SPI1_MOSI	GPIO	37	38	GPIO	I2S0_DIN	77	PJ.05
			GND	39	40	GPIO	I2S0_DOUT	78	PJ.06

Pines de Jetson Nano

En la Jetson Nano, el pin 8 (UARTT1_TXD) corresponde a la transmisión UART, mientras que el pin 10 (UARTT1_RXD) se utiliza para la recepción UART. Estos pines permiten establecer comunicación serial asincrónica con otros dispositivos externos. Además, el pin 6 se utiliza como conexión a tierra (GND), proporcionando una referencia común para la comunicación estable.

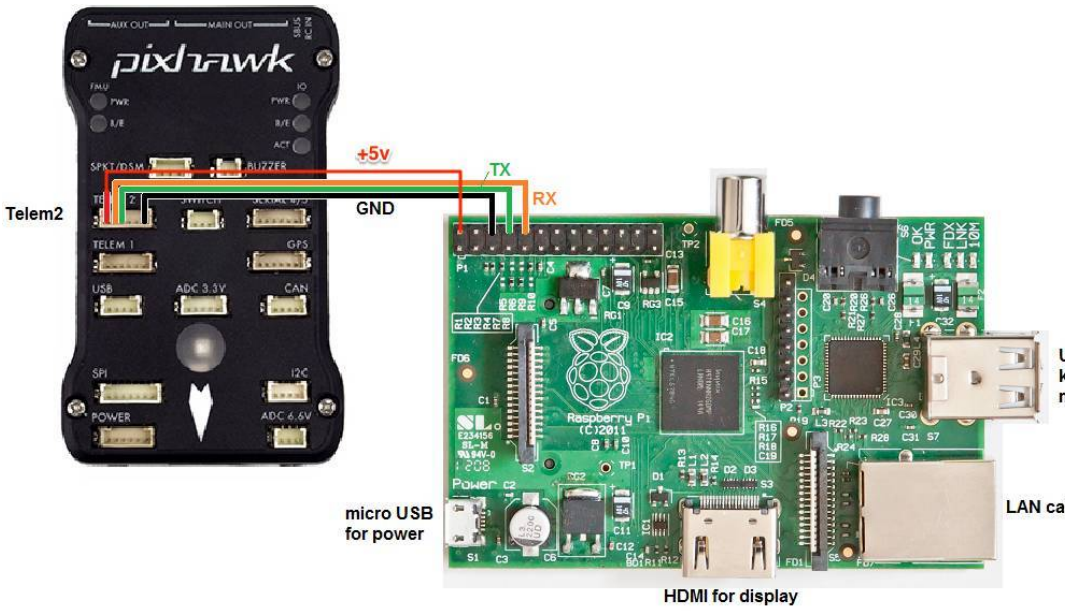
En la siguiente imagen se muestra como queda. Es necesario tener en cuenta que la alimentación no se realizará a través de esta conexión, por

lo que el cable rojo se puede obviar, ya que la controladora Pixhawk requiere un voltaje adicional para alimentar el GPS y otros módulos conectados a ella. La Jetson Nano tendrá su propia fuente de alimentación independiente.



Conexión UART Jerson Nano

El cable de conexión UART se conectará al puerto de telemetría 2 de la Pixhawk, asegurándose de verificar los pines de RX, TX y tierra correspondientes. Es importante tener en cuenta que los pines RX y TX pueden variar su posición entre los diferentes dispositivos, por lo que es necesario verificar la correspondencia correcta entre la Jetson Nano y la Pixhawk.



Esquema de conexión entre Pixhawk y Raspberry

En la imagen se muestra un esquema de conexión entre una Pixhawk y una Raspberry Pi. La conexión se establece a través de varios cables que conectan los pines GPIO de la Raspberry Pi con los pines correspondientes en la Pixhawk. Los detalles específicos de los pines utilizados en la Jetson Nano, que serían los explicados anteriormente, reemplazarían a los pines mencionados en la imagen. Tras hacer el cable y crimparlo queda así:

Conexión UART Pixhawk

Al establecer esta conexión correctamente, se logrará la comunicación entre la Jetson Nano y la Pixhawk, lo que permitirá intercambiar datos y comandos necesarios para el proyecto.

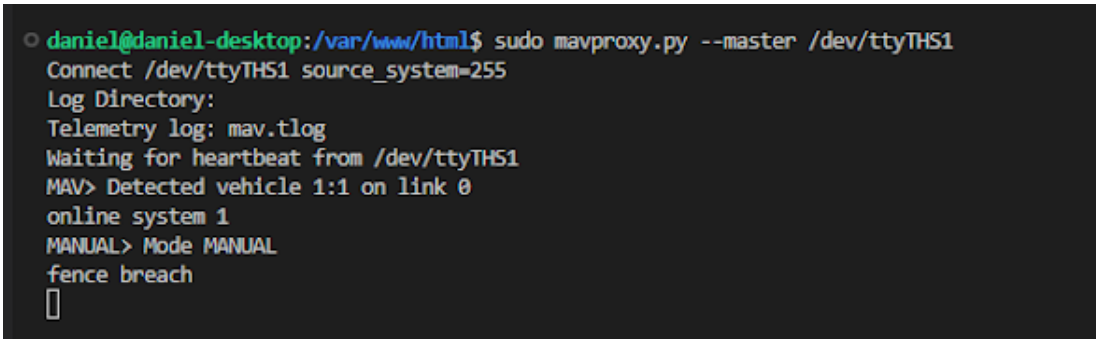
Utilizar MAVProxy es una opción para comprobar la conectividad entre la Jetson Nano y la Pixhawk. En el punto 5.3 se realizó la instalación de MAVProxy. Es importante tener en cuenta que la Pixhawk tiene una velocidad de baudios de 57600 y el puerto de comunicación UART en la Jetson Nano es /dev/ttyTHS1. Ambos dispositivos deben tener el mismo baud rate para establecer una comunicación exitosa. En caso de que no coincidan, es necesario ajustarlos para garantizar la compatibilidad.

Para comprobar la conexión, se ejecuta el siguiente comando en la terminal con el uso de "sudo" para acceder al puerto /dev/ttyTHS1:

```
Unset
sudo mavproxy.py --master /dev/ttyTHS1
```

Es importante mencionar que utilizar "sudo" permite al usuario acceder al puerto mencionado. Si se desea evitar el uso de "sudo", se pueden otorgar permisos al usuario correspondiente para acceder al puerto /dev/ttyTHS1.

Al ejecutar el comando, MAVProxy establecerá una conexión entre la Jetson Nano y la Pixhawk, lo que permitirá verificar la conectividad y realizar pruebas adicionales.



MAVProxy para comprobar la conexión

Y, como se puede observar, el modo actual es manual. Desde este punto, se tiene la capacidad de cambiar el modo de operación, enviar comandos para el despegue, cargar rutas de vuelo y realizar otras acciones relevantes. Esto proporciona un control completo sobre el dron y permite configurar su comportamiento según sea necesario.

5.6. MAVLink Comunicación entre Jetson Nano y Pixhawk(Mavsdk)

MAVLink es un protocolo de comunicación ligero y de código abierto utilizado en la industria de los drones y la robótica para la transmisión de datos y comandos entre sistemas aéreos no tripulados (UAVs) y estaciones de tierra.

En el proyecto desarrollado con la Jetson Nano, se ha utilizado el lenguaje de programación Python para implementar el programa de comunicación entre la Jetson Nano y la controladora del dron. Sin embargo, también es común utilizar C++ en proyectos similares, ya que es ampliamente utilizado en el ámbito de la robótica.

Existen diversas tecnologías y bibliotecas que utilizan el protocolo MAVLink. Algunas de las más utilizadas son:

-**pymavlink**: Es una biblioteca de Python que proporciona una interfaz completa para trabajar con mensajes MAVLink. Permite enviar y recibir mensajes MAVLink, así como acceder a los parámetros y realizar acciones en el vehículo.

-**Dronekit-Python**: Es una biblioteca de alto nivel para el desarrollo de aplicaciones de control de drones con Python. Utiliza MAVLink como protocolo de comunicación y ofrece una interfaz fácil de usar para interactuar con el vehículo.

-**MAVProxy**: Aunque MAVProxy es principalmente una herramienta de línea de comandos, también se puede utilizar como una biblioteca de Python para crear programas personalizados. Proporciona una interfaz para conectarse a vehículos aéreos, enviar y recibir mensajes MAVLink, y realizar acciones de control. En este proyecto se utilizó para comprobar la conectividad.

Entre estas opciones, se ha considerado el uso de Dronekit debido a su popularidad en proyectos similares que se encuentran en Internet. Sin embargo, hemos descubierto que Dronekit está un poco desactualizado y no brinda soporte completo para Python 3, según su página oficial y el repositorio oficial, donde se menciona la falta de personal para el mantenimiento continuo. A pesar de haber intentado instalarlo para realizar pruebas, nos hemos encontrado con errores de compatibilidad de versiones que han dificultado su funcionamiento.

Como alternativa, hemos optado por utilizar MAVSDK, una biblioteca desarrollada por la comunidad MAVLink. MAVSDK ofrece una interfaz sencilla y actualizada para interactuar con vehículos aéreos no tripulados a través de MAVLink.

MAVSDK es compatible con Python 3.7 y versiones posteriores. La última versión de Jetson-Inference que admite el reconocimiento de personas también es compatible con Python 3.7. Para utilizar ambas tecnologías en el mismo programa, es necesario instalar Python 3.7 y MAVSDK para Python 3.7. Esto permitirá aprovechar las funcionalidades de Mavsdk y Jetson-Inference en un programa integrado.

```
Unset
      /usr/bin/python3.7 -m pip install --upgrade pip
      /usr/bin/python3.7 -m pip install --upgrade setuptools
      /usr/bin/python3.7 -m pip install testresources

      /usr/bin/python3.7 -m pip install --no-cache-dir
--force-reinstall -Iv grpcio
```

Es importante instalar esto antes de poder instalar MAVSDK, ya que si lo hacemos directamente como dicen en la página oficial, en la Jetson Nano, ocurren varios errores, que se resuelven con los comandos anteriores. Y ahora si, se instala MAVSDK.

```
Unset
      /usr/bin/python3.7 -m pip install mavsdk
```

Con MAVSDK, esperamos tener una solución más estable y actualizada para la comunicación basada en MAVLink, lo que nos permitirá controlar e interactuar con el dron de manera efectiva utilizando Python como lenguaje de programación.

En el apartado 12 de la bibliografía se puede encontrar el enlace que proporciona MAVSDK con diferentes ejemplos. Para que el código funcione correctamente en la Jetson Nano conectada a la controladora del dron, es necesario realizar cambios en la parte de código relacionada con la configuración de la conexión. Los cambios deben reflejar los pines específicos de la Jetson Nano y la controladora del dron que se están utilizando. Ya que el siguiente código es para conectarse en local.

```
Python
         drone = System()
             await drone.connect(system_address="udp://:14540")

             print("Waiting for drone to connect...")
             async for state in drone.core.connection_state():
                 if state.is_connected:
                     print("Drone discovered!")
```

```
                    break
```

Quedará así:

```
Python
          drone = System()

              try:
                                                                    await
          asyncio.wait_for(drone.connect(system_address='serial:///dev/ttyTHS
          1:57600'), timeout=3)
              except asyncio.TimeoutError:
                   print("Error: No se pudo conectar al dron dentro del tiempo
          especificado")
                  file_path = "coordenadas.txt"  # Ruta del archivo
                  with open(file_path, "a") as file:
                      file.write("No se pudo conectar con el dron\n")
                  return

              print("Waiting for drone to connect...")
              async for state in drone.core.connection_state():
                  if state.is_connected:
                      print("Drone discovered!")
                      break
```

Este código fue desarrollado principalmente utilizando como referencia el enlace mencionado en el apartado 13 de la bibliografía, así como mis propios conocimientos y experiencia en el tema.

En este código, se crea una instancia de la clase System para representar el dron. Luego, se intenta establecer una conexión con el dron utilizando el método connect() de la instancia drone. Se especifica la dirección del sistema (system_address) como serial:///dev/ttyTHS1:57600, lo que indica una conexión UART con una velocidad de baudios de 57600.

Se utiliza await asyncio.wait_for() para esperar hasta que se establezca la conexión con el dron. Si no se puede establecer la conexión dentro del tiempo especificado (3 segundos en este caso), se muestra un

mensaje de error y se realiza un registro en un archivo de coordenadas indicando que no se pudo conectar con el dron.

Si se establece la conexión exitosamente, se entra en un bucle async for que recorre los estados de conexión del dron.

6. Pruebas con simulador

6.1. Objetivo

En el desarrollo del proyecto, se ha utilizado un simulador de drones debido a las ventajas que ofrece en comparación con las pruebas directas en el hardware del dron. Realizar pruebas en el dron real puede llevar mucho tiempo, especialmente cuando se trata de probar pequeñas funcionalidades ya que cada vez que se quisiera probar algo habría que agendar una cita con Javier.

Además, el hardware de los drones puede ser costoso y delicado. En caso de errores o problemas durante las pruebas en el dron real, existe el riesgo de dañar o incluso perder el equipo, lo cual implica un costo adicional y posibles retrasos en el proyecto.

6.2. Simulador utilizado

Existen varios simuladores de drones que puedes utilizar para simular y conectarlos a QGroundControl. Algunos de los simuladores más populares incluyen:

1. SITL (Software-in-the-Loop): Es parte de la suite de herramientas de Dronecode y permite simular drones utilizando el software ArduPilot. Puedes ejecutar SITL en tu computadora y conectarlo a QGroundControl para simular vuelos y realizar pruebas.
2. Gazebo: Es un popular simulador de robots utilizado en la robótica y la industria de drones. Gazebo proporciona un entorno de simulación 3D realista donde puedes simular drones y conectarlos a QGroundControl para realizar pruebas y simulaciones de vuelo.
3. AirSim: Es un simulador de drones desarrollado por Microsoft. AirSim ofrece una simulación precisa y realista de vuelo de

drones y proporciona una integración directa con QGroundControl, lo que te permite conectar y controlar los drones simulados en tiempo real.

4. JSBSim: Es una biblioteca de simulación de vuelo de código abierto que se utiliza para simular aeronaves, incluidos los drones. Puedes utilizar JSBSim para simular drones y conectarlos a QGroundControl para realizar pruebas y evaluaciones de rendimiento.

Estos son solo algunos ejemplos de simuladores de drones que puedes utilizar junto con QGroundControl.

Se ha elegido Gazebo como simulador de drones debido a su realismo, compatibilidad, herramientas de desarrollo, seguridad y capacidad para acelerar el tiempo de desarrollo. Utilizar Gazebo para las pruebas y simulaciones permite una validación exhaustiva del sistema y una optimización antes de implementarlo en un entorno real.

6.3. Gazebo

Para garantizar la compatibilidad entre Gazebo9 y QGroundControl, se ha decidido crear una máquina virtual con Ubuntu 20.04. Esto se debe a que Gazebo9 es compatible únicamente con versiones de Ubuntu 20.x.x en adelante, mientras que QGroundControl no es compatible con Ubuntu 22.04.

Esto se ha descubierto tras prueba y error, se ha llevado a cabo una prueba instalando Ubuntu 22.04 en una máquina virtual y se ha comprobado que Gazebo9 no funcionaba correctamente en esta versión. Debido a esta incompatibilidad, se ha optado por utilizar Ubuntu 18.04, pero se ha determinado que tampoco es una opción viable, ya que no es compatible con la versión de Gazebo9 requerida.

Por lo tanto, se ha tomado la decisión de crear una máquina virtual con Ubuntu 20.04 para garantizar la compatibilidad entre Gazebo9 y QGroundControl. Esta configuración permitirá realizar pruebas y simulaciones de manera efectiva utilizando ambos programas en un entorno coherente y funcional.

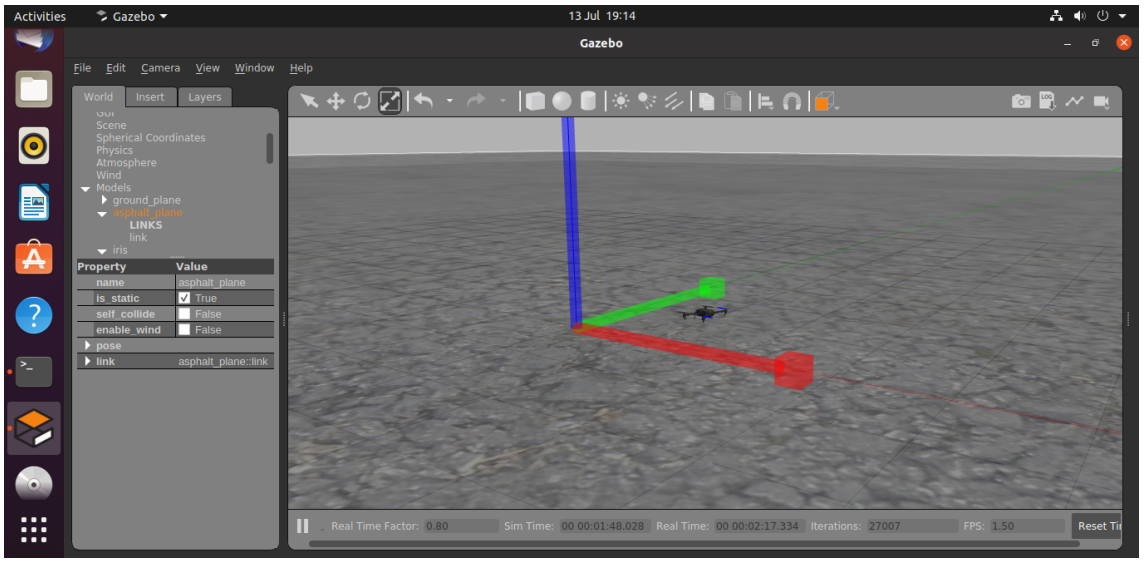
Una vez que se haya realizado la configuración inicial de la máquina virtual con Ubuntu 20.04, puedes abrir una terminal y seguir los pasos para

instalar Gazebo. Se detalla el procedimiento utilizando los pasos encontrados en **el enlace 18 de la bibliografía**.

Una vez instalado, se puede simular un Quadrotor simplemente con los siguientes comandos:

```
Unset
cd /path/to/PX4-Autopilot
make px4_sitl gazebo
```

Y tras ejecutarlo se puede ver como funciona correctamente.



Simulación de Quadrotor con Gazebo

La instalación de QGroundControl es un proceso sencillo y se puede realizar siguiendo los pasos que se indican en el enlace del punto 19 de la bibliografía.

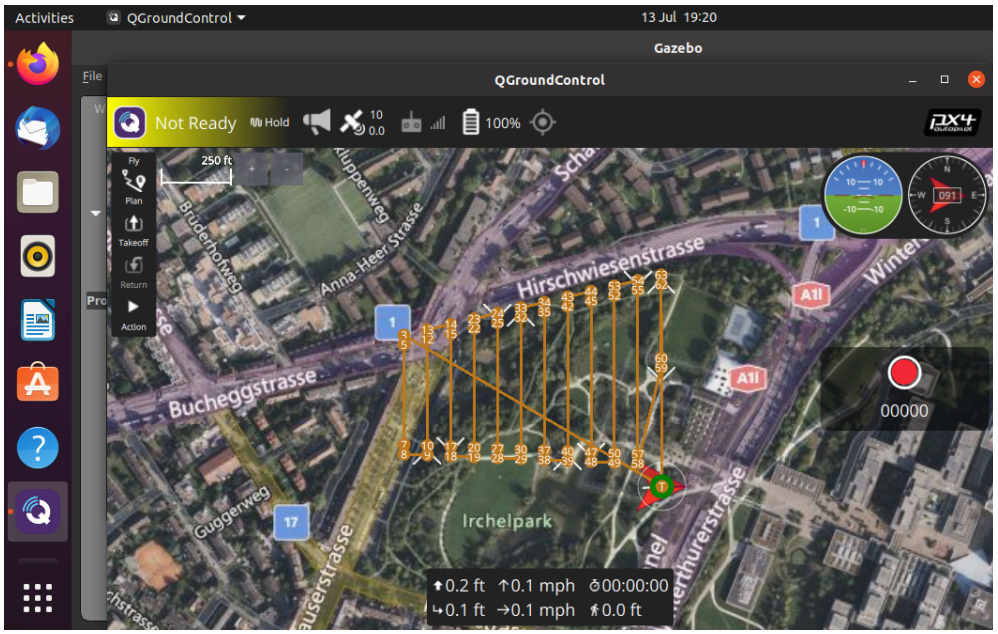
Una vez descargado, en la terminal, otorga permisos de ejecución al archivo QGroundControl.AppImage utilizando el siguiente comando:

```
Unset
chmod +x QGroundControl.AppImage
```

Después de haber dado los permisos necesarios, puedes ejecutar QGroundControl desde la terminal utilizando el siguiente comando:

```
Unset
./QGroundControl.AppImage
```

Esto iniciará QGroundControl y si se está ejecutando Gazebo, se conectará automáticamente al dron y se podrá comenzar a utilizar para interactuar con el dron y realizar las tareas necesarias.



QGroundControl conectado a Gazebo

6.4. Probando programas MAVSDK

Una de las principales razones por las que se optó por instalar un simulador fue para probar programas que utilizan el protocolo MAVLink para establecer conexiones y enviar comandos al dron simulado. Durante el desarrollo del proyecto, se realizaron pruebas utilizando diversas tecnologías, como pymavlink, dronekit y otras herramientas relacionadas con MAVLink.

Sin embargo, tras evaluar y experimentar con estas tecnologías, se ha decidido probar programas desarrollados con MAVSDK (MAVLink Software Development Kit). MAVSDK es una biblioteca de desarrollo que proporciona una interfaz unificada y simplificada para interactuar con vehículos que admiten el protocolo MAVLink.

Al probar programas desarrollados con MAVSDK en el simulador, se busca aprovechar estas ventajas y explorar las capacidades y funcionalidades que ofrece esta tecnología. Esto permite validar el funcionamiento de las aplicaciones, realizar pruebas exhaustivas y asegurar una integración adecuada con el dron simulado antes de pasar a las pruebas en un entorno real con un dron físico. En el apartado 20 de la bibliografía se encuentra el enlace al repositorio que tiene diferentes programas para probar en el simulador.

En el proyecto, se han utilizado programas para acceder a las coordenadas del dron y realizar reconocimiento de personas con Jetson-inference. Estos programas permiten obtener la ubicación precisa del dron y registrar las coordenadas. Además, cuando se reconoce a una persona, se guarda su ubicación y se captura una fotografía para asociarla a su identidad. Esto proporciona información valiosa para análisis y seguimiento posterior.

Asimismo, se ha implementado un programa adicional para permitir que el dron orbite alrededor de la persona detectada. Utilizando las coordenadas obtenidas, este programa establece un patrón de vuelo circular alrededor de la persona identificada. Esto brinda una perspectiva dinámica y posibilita capturar imágenes o videos desde diferentes ángulos.

A continuación se describe el programa que se conecta a un dron utilizando la biblioteca mavsdk y realiza una acción específica, que en este caso es realizar una órbita alrededor de un punto con una altura de 10 metros sobre el suelo. Aquí está la descripción paso a paso del programa:

1. Importación de bibliotecas:

```python
import asyncio from mavsdk
import System from mavsdk.action
import OrbitYawBehavior
```

- asyncio se utiliza para escribir código asincrónico.

- mavsdk es una biblioteca que proporciona una interfaz para comunicarse con drones utilizando el protocolo MAVLink.
- OrbitYawBehavior es una clase que representa el comportamiento de rotación alrededor de un punto en el espacio.

2. Definición de la función asincrónica run():

```python
async def run():
    drone = System()
    await drone.connect(system_address="udp://:14540")
    # Resto del código...
```

- Se crea una instancia de la clase System que representa al dron.
- Se establece una conexión con el dron utilizando await drone.connect() y se especifica la dirección del sistema como "udp://:14540". Esto indica que la conexión se realizará a través de UDP en el puerto 14540.

3. Espera hasta que el dron esté conectado:

```python
print("Waiting for drone to connect...")
async for state in drone.core.connection_state():
    if state.is_connected:
        print("Drone discovered!")
        break
```

- Se espera hasta que el estado de conexión del dron indique que está conectado.
- Una vez que se detecta que el dron está conectado, se imprime un mensaje indicando que el dron ha sido descubierto.

4. Espera hasta que el dron tenga una posición global estimada:

```python
print("Waiting for drone to have a global position estimate...")
async for health in drone.telemetry.health():
    if health.is_global_position_ok and health.is_home_position_ok:
        print("-- Global position estimate OK")
        break
```

- Se espera hasta que los parámetros de salud del dron indiquen que tiene una estimación de posición global válida.
- Si la estimación de posición global no es válida, se imprime un mensaje indicando que la estimación no es correcta.

5. Obtención de la posición actual del dron:

```python
async for position in drone.telemetry.position():
    orbit_height = position.absolute_altitude_m + 10
    break
```

- Se obtiene la posición actual del dron utilizando drone.telemetry.position().
- Se calcula la altura de la órbita sumando 10 metros a la altitud absoluta de la posición actual del dron.

6. Definición del comportamiento de rotación de la órbita:

```python
yaw_behavior = OrbitYawBehavior.HOLD_FRONT_TO_CIRCLE_CENTER
```

- Se define el comportamiento de rotación de la órbita como "HOLD_FRONT_TO_CIRCLE_CENTER", lo que significa que el dron mantendrá siempre la misma orientación hacia el centro de la órbita.

7. Realización de la órbita:

```python
print('Do orbit at 10m height from the ground')
await drone.action.do_orbit(radius_m=10,
```

```
                                velocity_ms=2,
                                yaw_behavior=yaw_behavior,
                                latitude_deg=position.latitude_deg,
                                longitude_deg=position.longitude_deg,
                                absolute_altitude_m=orbit_height)
```

- Se imprime un mensaje indicando que se realizará una órbita a una altura de 10 metros sobre el suelo.
- Se utiliza drone.action.do_orbit() para ejecutar la acción de órbita. Se especifican los siguientes parámetros:
    - radius_m: El radio de la órbita en metros.
    - velocity_ms: La velocidad de la órbita en metros por segundo.
    - yaw_behavior: El comportamiento de rotación de la órbita.
    - latitude_deg: La latitud del punto alrededor del cual se realizará la órbita.
    - longitude_deg: La longitud del punto alrededor del cual se realizará la órbita.
    - absolute_altitude_m: La altura absoluta a la cual se realizará la órbita.

8. Ejecución principal:

```python
if __name__ == "__main__":
    loop = asyncio.get_event_loop()
    loop.run_until_complete(run())
```

- Se verifica si el script se está ejecutando como programa principal.
- Se obtiene una instancia del bucle de eventos de asyncio.
- Se ejecuta la función run() utilizando loop.run_until_complete() para que se ejecute de forma asincrónica hasta que se complete.

En resumen, este programa se conecta a un dron utilizando mavsdk, espera hasta que esté conectado y tenga una estimación de posición

global válida, y luego realiza una acción de órbita alrededor de un punto específico con una altura de 10 metros sobre el suelo.

Estos programas trabajan de manera conjunta para brindar una solución completa en términos de acceso a las coordenadas del dron, reconocimiento de personas y almacenamiento de información asociada. Al utilizar las coordenadas obtenidas y la biblioteca Jetson-inference, se logra una integración efectiva entre la información espacial y los datos relacionados con las personas detectadas.

7. Página Web en Jetson Nano

Se ha creado una interfaz web en la Jetson Nano utilizando Apache 2 como servidor para mejorar la experiencia del usuario al ejecutar programas conectados por SSH. La interfaz web permite iniciar, detener y visualizar los resultados del programa de reconocimiento de personas, así como comprobar la conexión con la controladora. Esto simplifica la interacción y ofrece una forma más intuitiva de controlar y monitorear el sistema.

Para mantener la claridad del informe, no se explicará todo el código implementado para la página web. Sin embargo, en el repositorio asociado se pueden encontrar directorios con ejemplos de Dronekit, Mavsdk, jetson-infernece y pymavlink. Estos ejemplos proporcionan recursos útiles para comprender y utilizar estas bibliotecas en el contexto del proyecto.

Además, en la carpeta "pythonHTTPservers" del repositorio se encuentran los programas mod_wsgi que se ejecutan al pulsar los botones de la página web. Estos programas manejan las peticiones realizadas desde la interfaz y realizan diversas acciones, como comunicación con el dron y ejecución, detención del programa de reconocimiento de personas y contador de la cantidad de imágenes que se han reconocido, ya que directamente desde el index.html no se podía hacer por temas de seguridad.

Si se desea obtener más información sobre el funcionamiento de estos programas mod_wsgi y su integración con la página web, se recomienda revisar el repositorio y explorar los directorios correspondientes.

7.1. Apache2

Para instalar y comenzar Apache2, sigue estos pasos:

1. Actualizar los repositorios ejecutando el comando:

```
Unset
sudo apt-get update
```

2. Instalar Apache2 ejecutando el comando:

```
Unset
sudo apt-get install apache2
```

3. Iniciar el servicio de Apache2 con el comando:

```
Unset
 sudo service apache2 start
```

4. Se puede verificar el estado del servicio ejecutando:

```
Unset
sudo service apache2 status
```

Una vez que Apache2 esté en funcionamiento, se puede acceder a la página web predeterminada ingresando la dirección IP de la Jetson Nano en un navegador web. Al conectarse al enrutador de la Jetson Nano, esta ha recibido la dirección IP 192.168.1.100 en esa red (según se explica en el punto 8.5), simplemente se ingresa esa dirección IP en el navegador y se abrirá la página web predeterminada de Apache.

Para crear la página web, hay que eliminar el archivo de ejemplo que se encuentra en el directorio /var/www/html y reemplazarlo con la página que se quiere ejecutar. Hay que asegurarse de tener un archivo index.html en el directorio /var/www/html con el contenido que se desea mostrar en la página web. Al hacer esto, la página web que se ha creado se ejecutará y estará disponible cuando ingreses la dirección IP de la Jetson Nano en el navegador.

7.2. mod-wsgi

El módulo mod_wsgi es una extensión del servidor web Apache2 que permite ejecutar aplicaciones web escritas en Python de manera eficiente y segura. Proporciona una interfaz entre Apache2 y la aplicación web Python, lo que facilita su despliegue y ejecución en un entorno de servidor web.

Mod_wsgi permite que las aplicaciones web Python se ejecuten como procesos independientes dentro del servidor Apache2, lo que proporciona beneficios significativos en términos de rendimiento y escalabilidad. Además, el módulo mod_wsgi ofrece una gestión eficiente de la carga y el rendimiento, lo que permite manejar múltiples solicitudes concurrentes de manera simultánea.

En este proyecto, se utiliza el módulo mod_wsgi en Apache2 para ejecutar programas de reconocimiento de personas escritos en Python y comprobar la conectividad. Esto se realiza a través de una página web interactiva con botones que permiten ejecutar, detener y verificar la conectividad. El módulo mod_wsgi se encarga de gestionar las solicitudes y ejecutar los scripts de Python correspondientes. En resumen, el proyecto combina el uso de mod_wsgi en Apache2 y una página web para controlar y monitorizar los programas de reconocimiento de personas y la conectividad.

En el punto 23 de la bibliografía se puede ver paso a paso como instalarlo y un ejemplo.

7.3. Permisos

En mod-wsgi el usuario que ejecuta los programas python es www-data. Para configurar adecuadamente los permisos y el usuario www-data en la Jetson Nano, sigue estos pasos:

1. Otorga permisos a la carpeta /var/www/html para que el usuario www-data pueda acceder a ella utilizando el comando:

```
Unset
sudo chmod 777 /var/www/html
```

Esto permitirá el acceso completo a la carpeta por parte de todos los usuarios.

2. Antes de modificar el usuario www-data, asegurarse de tener privilegios de superusuario. Para lograrlo, ejecutar el siguiente comando:

```
Unset
sudo visudo
```

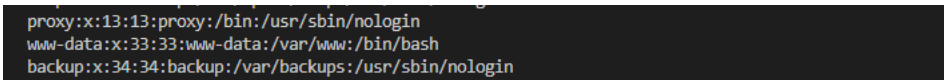
Después de introducir la contraseña de tu usuario entraremos en el fichero sudoers (siempre desde el editor que tenemos definido por defecto). Hay que añadir debajo de la línea:

```
Unset
root    ALL=(ALL:ALL) ALL
user    ALL=(ALL:ALL) ALL
```

Siendo user el usuario al que queremos dar privilegios.

3. Una vez que se tenga privilegios de superusuario, se puede cambiar el shell del usuario www-data para permitirle utilizar bash en lugar de un shell limitado de servicio. La información sobre cómo asignar un nuevo shell de bash en el enlace proporcionado en el punto 24

de la bibliografía. Tras hacer esto podemos ver que este usuario tiene asignada una bash.



WWW-data con bash asignada

4. Cambia la contraseña del usuario www-data utilizando el comando:

```
Unset
sudo passwd www-data
```

A continuación, se solicita que se introduzca una nueva contraseña para el usuario www-data. Se introduce la que se ha utilizado en el proyecto 'Z7ZhekVI'

5. Para permitir que el usuario www-data pueda ejecutar programas como otro usuario (por ejemplo, daniel), hay que agregarlo al grupo sudo o admin. Esto se hace como se explicó en el punto 2 de este apartado.

Una vez completados estos pasos, los permisos y la configuración del usuario www-data estarán actualizados y se podrán ejecutar programas y acceder a la shell como se ha especificado.

7.4. Página Web

La página web ha sido desarrollada utilizando Bootstrap para garantizar un formato adaptable y funcional en diferentes dispositivos, como ordenadores, móviles y tablets. El diseño se ajusta automáticamente para aprovechar el espacio de pantalla disponible y ofrece una apariencia coherente y organizada. Además, se utilizan características y componentes adicionales de Bootstrap para mejorar la experiencia del usuario, como una navegación intuitiva y elementos interactivos. En resumen, Bootstrap permite que la página web se vea y funcione de manera óptima en cualquier dispositivo. En el ropsitorio del punto 26 de la bibliografía se encuentra el enlace para el repositorio que contiene la página web. Para

hacer que funcione correctamente en la jetson nano hay que realizar una serie de pasos:

Para configurar un entorno web utilizando el módulo mod_wsgi y un repositorio de GitHub. Los pasos incluyen:

1. Acceder a la carpeta /var/www/html y eliminar su contenido
2. Enlazar el repositorio de GitHub

‘https://github.com/Daniel-Arbelo/web-tfg-mod_wsgi/tree/main ‘

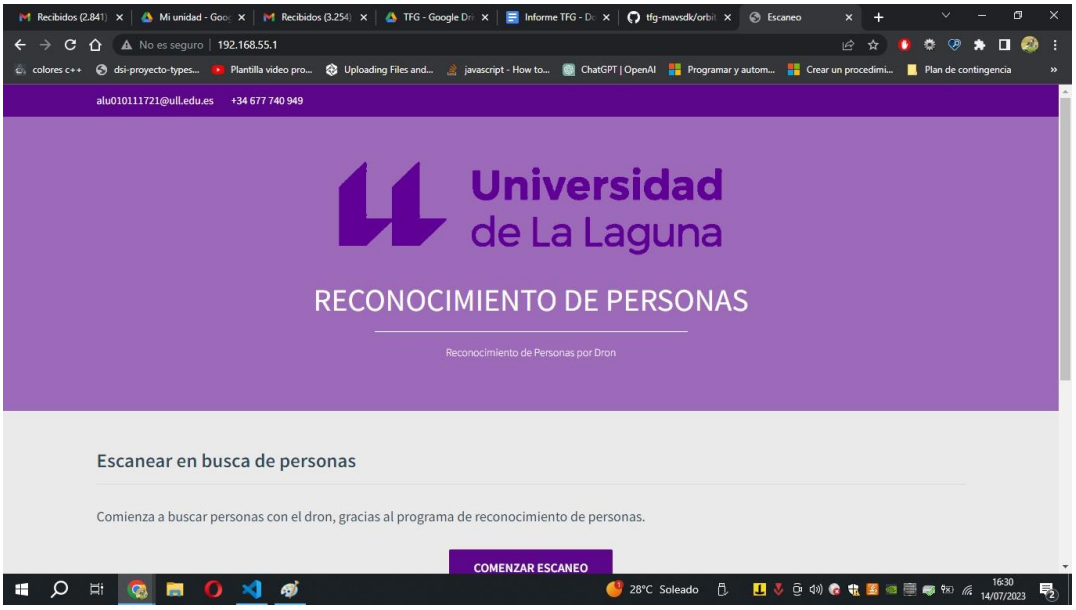
Y se hacer un pull.

Después de esto solo queda configurar mod_wsgi, para esto en la carpeta /etc/apache2/conf-available se crea el archivo mod-wsgi.conf con una configuración específica. Que es la siguiente:

```
WSGIScriptAlias /ejecutar_programa
/var/www/html/pythonHTTPservers/ejecutar-programa.py
WSGIScriptAlias /contador_archivos
/var/www/html/pythonHTTPservers/contador-archivos.py
WSGIScriptAlias /terminar_proceso
/var/www/html/pythonHTTPservers/detener-programa.py
WSGIScriptAlias /ejecutar_programa_coordenadas
/var/www/html/pythonHTTPservers/info-controladora.py
```

Después de esto ya se puede acceder a la página y su funcionalidades.

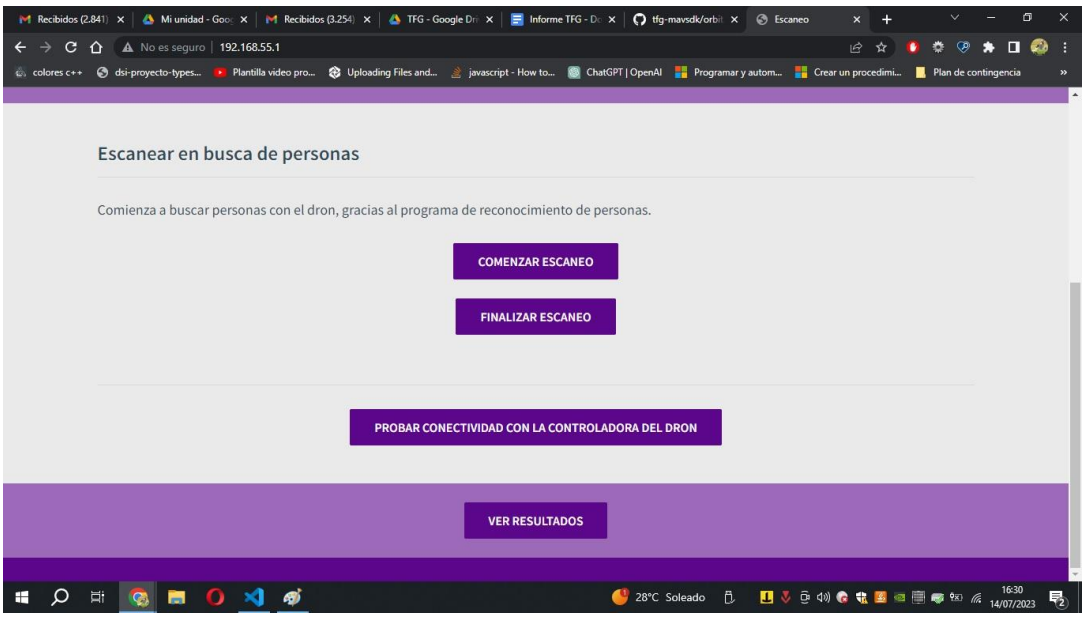
Al abrir la página se ve lo siguiente:



Página web

En la página, se encuentran varios botones que realizan diferentes funciones relacionadas con el programa de reconocimiento de personas:

1. El primer botón ejecuta el programa de reconocimiento de personas: Al hacer clic en este botón, se activa el programa de reconocimiento de personas, que se explica en el siguiente punto (7.5)
2. El segundo botón finaliza el escaneo: Este botón detiene el proceso haciendo un kill del pid del programa.
3. El tercer botón comprueba la conexión con el dron: Este botón verifica la conectividad y la comunicación con un dron por el puerto /dev/ttyTHS1 que es el que se utiliza para la conexión UART.
4. El último botón muestra las imágenes de las personas detectadas con la ubicación debajo de ellas: Al hacer clic en este botón, se despliegan las imágenes de las personas que han sido detectadas por el programa de reconocimiento. Estas imágenes pueden ser visualizadas junto con información adicional, como la ubicación, si se ha conectado al dron y acceder a ella.



Funcionalidades de la página

7.5. Programa para controlar dron con MAVSDK al reconocer personas

El programa llamado camara-detection.py en el repositorio es el programa principal. Para leer este apartado y entenderlo es mejor tener el código abierto, se encuentra en el repositorio enlazado en el punto 26 de la Bibliografía.

El código proporcionado es una implementación de un programa que utiliza la biblioteca Jetson Inference para la detección de personas en tiempo real en un video proveniente de una fuente de video. Se sabe que la webcam es /dev/video0, porque se ejecuta el siguiente comando, para saber qué dispositivos disponibles hay, si se quisiera utilizar otra cámara, habría que conectarla a la jetson nano y mirar como se llama con este comando para ver los dispositivos de video conectados.

En primer lugar se instala:

```
Unset
        sudo apt-get install v4l-utils
```

Y para saber qué dispositivos hay, se ejecuta:

```
Unset
        v4l2-ctl --list-devices
```

Además, se integra con la biblioteca MAVSDK para acceder y controlar un dron.

Este es el flujo general del código:

1. Importación de módulos y paquetes necesarios: El código comienza importando los módulos y paquetes requeridos, incluyendo "detectNet" de Jetson Inference, "videoSource" y "videoOutput" de Jetson Utils, y otros paquetes relacionados con la comunicación del dron.
2. Creación de instancias y configuración: Se crea una instancia de "detectNet" utilizando el modelo "ssd-mobilenet-v2" con un umbral de detección establecido en 0.5. Luego, se crea una instancia de "videoSource" para capturar el video de la cámara en /dev/video0. Además, se crea una instancia de "videoOutput" para guardar las imágenes resultantes de la detección de las personas detectadas.
3. Limpieza de capturas anteriores: Se ejecuta un comando de sistema para eliminar todas las imágenes previas almacenadas en el directorio "/var/www/html/imagenesPasadasPrograma/".
4. Configuración inicial de archivos de salida: Se sobrescribe el contenido del archivo "/var/www/html/coordenadas.txt" con una cadena vacía para prepararlo para las nuevas coordenadas.
5. Función "detect_people": Esta función se ejecuta de forma asíncrona en un bucle infinito. Captura imágenes de la fuente de video y las procesa utilizando la red neuronal "detectNet" para detectar personas. Si se detecta una persona, se guarda la imagen utilizando "videoOutput", se obtienen las coordenadas del dron mediante la función "get_drone_coordinates", y se almacenan en el archivo de coordenadas.
6. Función "get_drone_coordinates": Esta función se ejecuta de forma asíncrona para obtener las coordenadas del dron utilizando la biblioteca MAVSDK. Se establece una conexión con el dron, se esperan las condiciones necesarias (conexión y

estimación de posición global), se obtienen las coordenadas y se almacenan en el archivo de coordenadas. Además, se realiza una acción de órbita alrededor de la ubicación del dron a una altura de 10 metros.

7. Punto de entrada y ejecución principal: Se obtiene el bucle de eventos de asyncio y se crean tareas para ejecutar las funciones "detect_people" y "get_drone_coordinates". Luego, se ejecuta el bucle de eventos hasta que todas las tareas se completen.

En resumen, este código combina la detección de personas en tiempo real utilizando Jetson Inference con la comunicación y el control de un dron utilizando MAVSDK. Proporciona una integración de detección de personas y funcionalidades de dron en un programa que puede ejecutarse de manera continua.

8. Hacer Jetson Nano de puente entre Pixhawk y estaciones de tierra

8.1. Objetivo

El objetivo es montar una Jetson Nano en un dron y conectarla a la Pixhawk mediante una conexión UART. Además, la Jetson Nano estará conectada a un enrutador. El propósito es permitir que cuando el ordenador del operario de tierra se conecte a esta red Wi-Fi, la Jetson Nano actúe como puente para reenviar la información del controlador de vuelo. De esta manera, al abrir programas como QGround Control o Mission Planner, el operario podrá reconocer el dron. Se han investigado diversas tecnologías para lograr esto.

Algunos proyectos han utilizado Dronekit, pero, como se mencionó anteriormente, no se pudo hacer que funcionara correctamente. Otra opción es utilizar MAVProxy, aunque existen tecnologías específicas para Raspberry Pi u otros tipos de ordenadores. Después de una investigación exhaustiva, se ha encontrado MAVLink Router, que resulta ser la solución

ideal para el propósito deseado. Sin embargo, es importante tener en cuenta algunos pasos para que todo funcione correctamente.

El objetivo es reenviar los paquetes UART a las direcciones IP de los dispositivos conectados a esta red.

8.2. Seguridad

Con respecto a la seguridad, es crucial establecer una contraseña robusta en el enrutador para evitar que cualquier persona pueda tomar el control del dron de forma no autorizada. Además, se limita el reenvío de los paquetes UART a solo dos direcciones IP, asignadas a los primeros dos dispositivos que se conecten a esta red.

Estas medidas de seguridad garantizan que sólo los dispositivos autorizados puedan recibir los paquetes UART y, por lo tanto, tener acceso a la información del controlador de vuelo del dron. Al limitar el número de dispositivos permitidos y asegurar la red mediante una contraseña segura, se reduce significativamente el riesgo de intrusiones no deseadas y se protege la integridad del sistema.

Es importante destacar que la implementación de medidas de seguridad adicionales, como cifrado de datos y autenticación de dispositivos, también puede ser beneficiosa para garantizar la protección completa de la comunicación y evitar cualquier posible compromiso de seguridad. La seguridad es un aspecto esencial en cualquier proyecto tecnológico, y aunque pueda no ser factible implementar todas las medidas en este momento, es fundamental tenerlo presente y considerarlo en futuros desarrollos.

8.3. Instalación y configuración

La página oficial de MAVLink Router se puede encontrar en el punto 14 de la bibliografía, donde se proporciona información detallada y recursos sobre esta herramienta. Allí se encuentran enlaces y documentación relevante para su uso.

En el punto 15 de la bibliografía, se describen los pasos para instalar MAVLink Router en la Jetson Nano. Siguiendo estos pasos, no ha habido problemas durante la instalación. A continuación, se procederá a explicar la configuración necesaria para su correcto funcionamiento.

Para configurar la redirección en MAVLink Router, se procede a crear un archivo de configuración llamado "main.conf" en la carpeta /etc/mavlink-router. Este archivo contendrá los parámetros necesarios para llevar a cabo la redirección adecuadamente.

El archivo de configuración final queda de la siguiente manera:

```
daniel@daniel-desktop:/var/www/html$ cat /etc/mavlink-router/main.conf
[General]
   TcpServerPort=5760
   ReportStats=false
   MavlinkDialect=common

[UartEndpoint ttyTHS1]
   Device=/dev/ttyTHS1
   Baud=57600

[UdpEndpoint groundStation]
   Mode=Normal
   Address=192.168.1.101
   Port=14550

[UdpEndpoint groundStation2]
   Mode=Normal
   Address=192.168.1.102
   Port=14550
daniel@daniel-desktop:/var/www/html$
```

Archivo de configuración de MAVLink Router

En la sección [General], se especifican configuraciones generales como el puerto del servidor TCP, la opción de informar estadísticas (ReportStats) y el dialecto MAVLink utilizado (MavlinkDialect).

Luego, en la sección [UartEndpoint ttyTHS1], se configura el endpoint UART para la comunicación con el puerto /dev/ttyTHS1 a una velocidad de baudios de 57600.

A continuación, se definen dos endpoints de UDP en las secciones [UdpEndpoint groundStation] y [UdpEndpoint groundStation2]. Estos endpoints se configuran para comunicarse con las direcciones IP 192.168.1.101 y 192.168.1.102, respectivamente, en el puerto 14550. Estos endpoints se utilizan para redirigir los mensajes MAVLink hacia las estaciones terrestres conectadas.

En resumen, el archivo de configuración final establece los parámetros necesarios para la comunicación a través de UART y UDP,

definiendo las direcciones IP y puertos correspondientes para las estaciones terrestres.

Ahora solo queda ejecutar el comando

```
Unset
sudo /usr/bin/mavlink-routerd
```

iniciará el proceso de MAVLink Router y comenzará a funcionar según la configuración establecida.

Si deseas agregar más dispositivos para la redirección de los paquetes MAVLink, modificar el archivo de configuración "main.conf" y agregar nuevas secciones para cada dispositivo adicional. Por ejemplo:

```
Unset
[UdpEndpoint groundStation3]
Mode=Normal
Address=192.168.1.103
Port=14550
```

Una vez completados estos pasos, MAVLink Router comenzará a redirigir los paquetes MAVLink a las nuevas direcciones IP especificadas para los dispositivos adicionales.

8.4. Servicio de MAVLink Router

Para evitar tener que ejecutar manualmente el comando "sudo /usr/bin/mavlink-routerd" cada vez que se enciende la Jetson Nano, se crea un servicio que inicie automáticamente la redirección al encender el dispositivo. Esto se logra creando un archivo en la ubicación "/etc/systemd" llamado "mavlink-router.service". Que contiene lo siguiente:

```
daniel@daniel-desktop:/etc/systemd$ cat mavlink-router.service
[Unit]
Description=MAVLink Router
Wants=zerotier-one.service
After=network-online.target zerotier-one.service

[Service]
Type=simple
ExecStart=echo 'Z7ZhekVI' | sudo /usr/bin/mavlink-routerd
Restart=on-failure
RestartSec=3

[Install]
WantedBy=multi-user.target
daniel@daniel-desktop:/etc/systemd$
```

Archivo de configuración del servicio MAVLink Router

Ahora solo queda empezar el servicio "mavlink-router" ejecutando el siguiente comando:

```
Unset
sudo systemctl start mavlink-router.service
```

Este comando iniciará el servicio y la redirección de MAVLink Router comenzará a funcionar automáticamente.

Además, se deseas habilitar el servicio para que se inicie automáticamente en cada arranque del sistema, ejecutar el siguiente comando:

```
Unset
sudo systemctl enable mavlink-router.service
```

Esto asegurará que el servicio "mavlink-router" se inicie de forma automática cada vez que enciendas la Jetson Nano.

Una vez ejecutados estos comandos, el servicio estará en funcionamiento y configurado para iniciarse automáticamente, lo que te permitirá ahorrar tiempo y garantizar que la redirección de MAVLink Router esté siempre activa y lista para su uso.

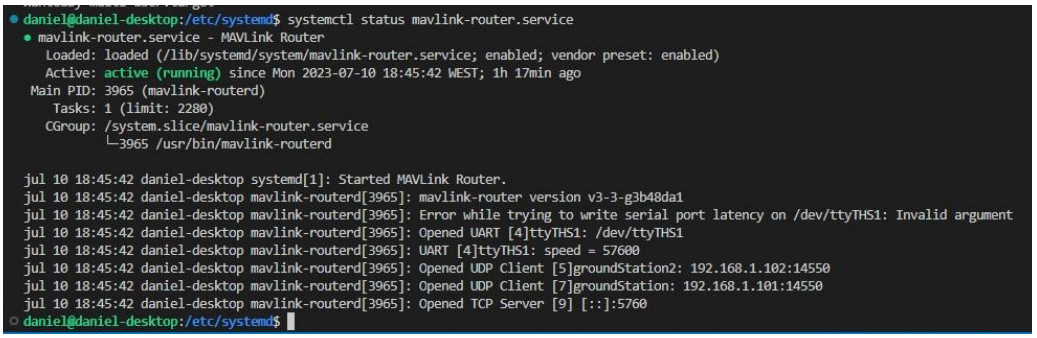
Si se necesita detener el servicio en algún momento, puedes utilizar el comando:

```
Unset
sudo systemctl stop mavlink-router.service
```

Para comprobar que está activo basta con ejecutar el siguiente comando:

```
Unset
systemctl status mavlink-router.service
```

Que dá el siguiente resultado:



Estado del servicio de MAVLink Router

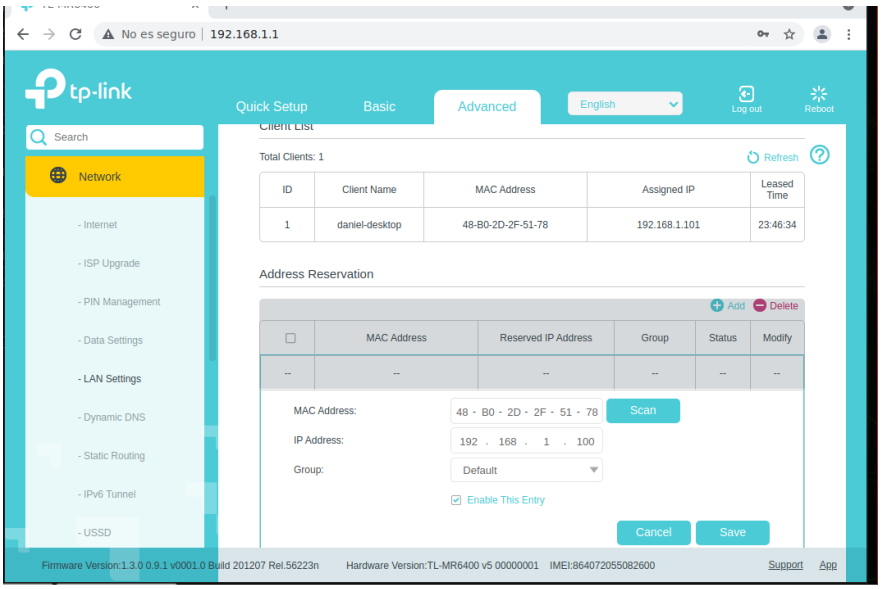
8.5. Asignar la misma IP a la Jetson Nano siempre

Para garantizar un funcionamiento estable de la redirección en la Jetson Nano, es necesario asignarle una dirección IP estática en el enrutador. Esto evita problemas de asignación automática de direcciones IP y garantiza que la Jetson Nano siempre tenga la misma dirección IP deseada, en este caso, 192.168.1.100. Ya que si no se le puede asignar esta dirección a otro dispositivo que se conecte a esta red. Y habrá momentos en los que fallará.

A continuación, se presenta un ejemplo de cómo realizar esta configuración en un enrutador TP-Link, como se menciona en el punto 16 y 17 de la bibliografía:

1. Conectarse a la Jetson Nano y abrir un navegador web.
2. En la barra de direcciones del navegador, ingresar la dirección IP del enrutador TP-Link. Por lo general, la dirección IP predeterminada es 192.168.0.1 o 192.168.1.1. Se puede verificar esta información en el manual del enrutador o en la parte inferior o trasera del dispositivo. Por ejemplo, si es 192.168.1.1, ingresa esa dirección.
3. Ingresar la contraseña de administrador del enrutador. Si no se ha cambiado la contraseña previamente, se puede encontrar en el manual del enrutador o en la etiqueta del dispositivo.
4. Una vez que se haya iniciado sesión en la configuración del enrutador, buscar la sección "Advanced"(Avanzado) y luego "Network"(Red) .
5. Dentro de la sección de configuración de red, buscar "LAN Setting" (Configuración de LAN).
6. Desplazarse hacia abajo hasta encontrar "Address Reservation" (Reserva de direcciones). Aquí se agrega la dirección MAC de la Jetson Nano y la dirección IP que se desea asignar de forma estática.
7. Después de agregar la dirección MAC y la dirección IP, guardar los cambios realizados y reiniciar el enrutador para que la configuración tenga efecto.

Una vez completados estos pasos, la Jetson Nano recibirá la dirección IP deseada (192.168.1.100) cada vez que se conecte al enrutador. Esto asegurará que MAVLink Router funcione correctamente al tener una dirección IP estática asignada a la Jetson Nano.



Asignar IP estática a Jetson Nano

8.Conclusiones

El uso de drones en este proyecto permite una perspectiva aérea que puede ser especialmente beneficiosa para la detección y el seguimiento de personas en áreas extensas o de difícil acceso. La integración de algoritmos y técnicas de visión por computadora para el reconocimiento de personas permite identificar y rastrear a las personas en las imágenes o videos capturados por el dron.

Una vez que se detecta a una persona, el dron guarda su ubicación para realizar un seguimiento preciso. A continuación, el dron orbita alrededor de la persona detectada, lo que le permite obtener una visión detallada desde diferentes ángulos. Durante este recorrido, el dron puede recopilar información adicional, como imágenes de alta resolución. Esta información puede resultar valiosa para aplicaciones como vigilancia, investigación o análisis de patrones de comportamiento.

Es importante tener en cuenta que la seguridad y la privacidad son consideraciones clave en este proyecto. Es necesario cumplir con las regulaciones aplicables y garantizar la protección de los datos de las personas involucradas en el proceso de reconocimiento. Además, se deben implementar medidas adecuadas para garantizar la seguridad de las operaciones con drones y evitar cualquier riesgo potencial.

En resumen, el proyecto muestra un enfoque innovador y prometedor. Combina tecnologías avanzadas para obtener datos valiosos desde una perspectiva aérea, con aplicaciones potenciales en áreas como búsqueda y rescate, seguridad, monitoreo de eventos y análisis de audiencia. Sin embargo, es fundamental abordar las consideraciones de seguridad, privacidad y cumplimiento normativo para garantizar un desarrollo responsable y ético de este proyecto.

9. Líneas Futuras.

Una ampliación podría estar relacionada con la interacción y comunicación. Sería interesante habilitar una comunicación bidireccional entre el dron y las personas detectadas. Esto podría lograrse a través del uso de altavoces o pantallas, permitiendo al dron proporcionar instrucciones o información a las personas. Asimismo, las personas podrían interactuar con el dron mediante comandos de voz o gestos, lo que abriría posibilidades de colaboración y control compartido.

Otra posible ampliación es la integración del proyecto con sistemas de seguridad existentes. Esto implicaría conectar el dron y su capacidad de reconocimiento de personas con sistemas de alarma o cámaras de vigilancia ya implementados. De esta manera, se podría lograr una respuesta rápida y automatizada ante la detección de comportamientos sospechosos o amenazantes, mejorando así la seguridad en diversos entornos.

Es importante destacar que al ampliar el proyecto, se debe considerar las implicaciones éticas, legales y de privacidad correspondientes.

Bibliografía

1 Interferencias

https://forums.developer.nvidia.com/t/jetson-nano-causes-massive-interference-to-gps-module-beside-it/205208/11

2 Soportes Jetson Nano

https://www.dropbox.com/sh/ip13e0wacflxtbb/AADVNoiI6ry5AH1D5ktzPIPRa?dl=0

3 Instalación de sistema operativo Jetson Nano

https://developer.nvidia.com/embedded/learn/get-started-jetson-nano-2gb-devkit

4 Repositorio de jetson-inference

https://github.com/dusty-nv/jetson-inference

5 Tutorial de youtube de detección de objetos en tiempo real(utilizando jetson-inference)

https://www.youtube.com/watch?v=bcM5AQSAzUY&t=83s&ab_channel=NVIDIADeveloper

6 Jetson-inference building the project from source

https://github.com/dusty-nv/jetson-inference/blob/master/docs/building-repo-2.md

7 Discusión sobre versiones de python para jetson-inference

https://github.com/dusty-nv/jetson-inference/issues/936

8 Pines de Jetson Nano

https://developer.nvidia.com/embedded/learn/jetson-nano-2gb-devkit-user-guide

9 Conexión entre Pixhawk y Raspberry

https://discuss.ardupilot.org/t/raspberry-pi-3-connecting-to-pixhawk/9524/3

10 Video para comprobar la conectividad entre la Jetson Nano y la Pixhawk

https://www.youtube.com/watch?v=nIuoCYauW3s&t=6s&ab_channel=aka%3AMatchstic

11 MAVSDK-Python

https://github.com/mavlink/MAVSDK-Python

12 MAVSDK examples

https://github.com/mavlink/MAVSDK-Python/tree/main/examples

13 Conectarse a drone MAVSDK

https://mavsdk.mavlink.io/main/en/cpp/guide/connections.html

14 MAVLink Router

https://github.com/mavlink-router/mavlink-router

15 Instalar MAVLink Router

https://bellergy.com/6-install-and-setup-mavlink-router/

16 Asignar ip estática a dispositivo en router tplink

https://www.tp-link.com/us/support/faq/560/

17 Asignar ip estática a dispositivo en router tplink

https://www.tp-link.com/us/support/faq/182/#:~:text=Click%20DHCP%2D%3EAddress%20Reservation%20on,and%20click%20Add%20New%20button.&text=Input%20the%20MAC%20address%20and,be%20reserved%20by%20the%20rout

18 Instalar Gazebo

https://docs.px4.io/v1.12/en/simulation/gazebo.html

19 Descargar QGroundControl

https://docs.qgroundcontrol.com/master/en/getting_started/download_and_install.html

20 Programas MAVSDK

https://github.com/Daniel-Arbelo/tfg-mavsdk/settings

21 Instalar Mission Planner

https://ardupilot.org/planner/docs/mission-planner-installation.html

22 Instalar y comenzar Apache 2

https://www.youtube.com/watch?v=l6de3crYv_k&ab_channel=TECHDHEE

23 Instalar mod_wsgi en Ubuntu

https://ubiq.co/tech-blog/install-mod_wsgi-ubuntu/

24 Asignarle bash a un usuario

https://protegermipc.net/2021/06/01/solucion-al-problema-this-account-is-currently-not-available-en-linux/

25 Darle permisos de administrador a usuario

https://slimbook.es/tutoriales/linux/86-anadir-usuario-al-fichero-sudoers

26 Repositorio con página web del proyecto

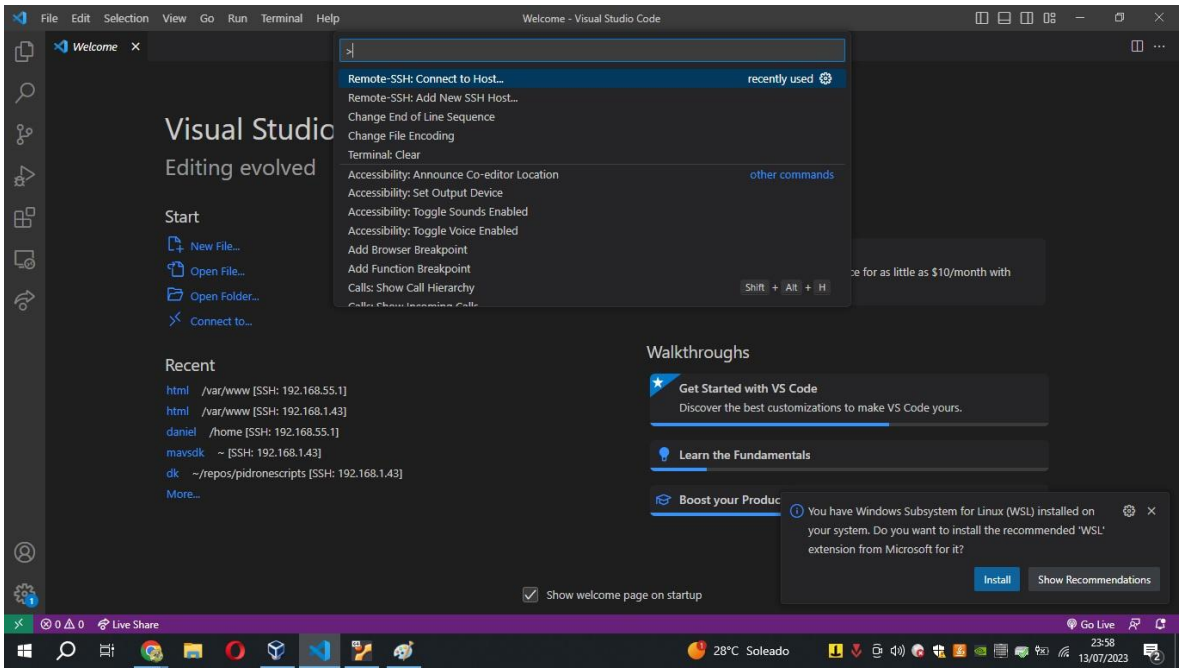
https://github.com/Daniel-Arbelo/web-tfg-mod_wsgi/tree/main

Anexos

Anexo 1 - Conexión por SSH mediante visual studio code

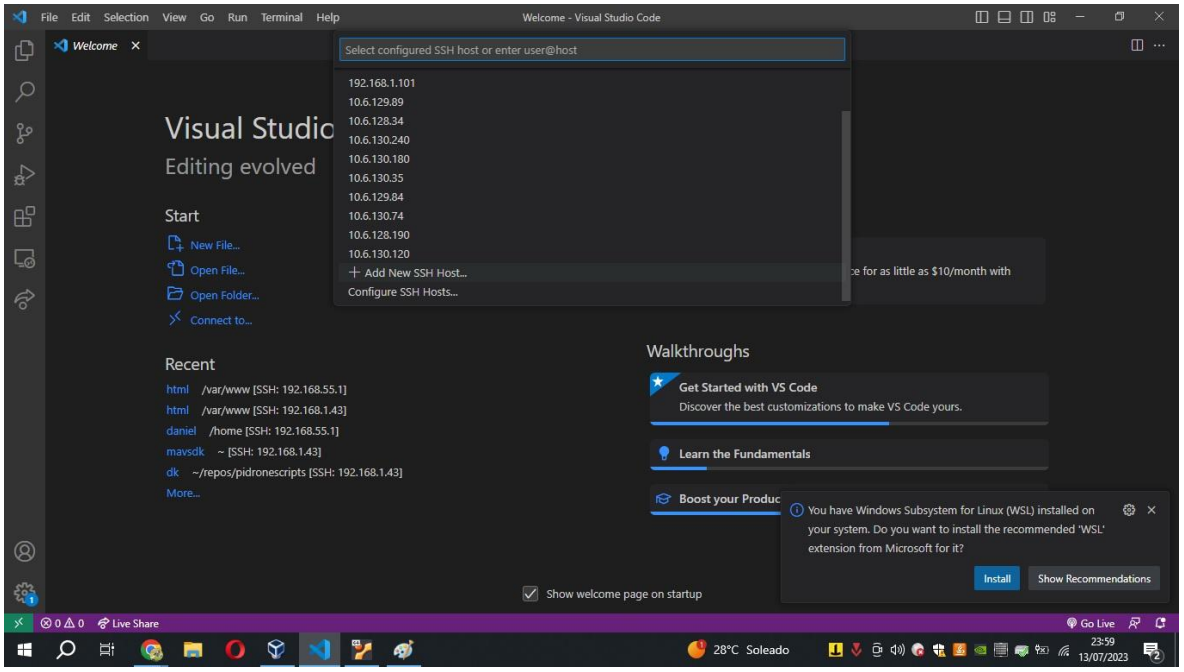
Para conctarse por SSH a la Jetson Nano y configurarla, hay algunos pasos que debes seguir. En primer lugar, necesitarás instalar la extensión "Remote - SSH" en el entorno de desarrollo. Una vez que la extensión esté instalada, se pude proceder a establecer la conexión.

1. Abrir un entorno de desarrollo y presionar Ctrl + P para abrir el buscador.
2. Escribir ">Remote-SSH: Connect to Host" en el buscador y seleccionar la opción correspondiente.



Remote-SSH:Connect to Host

3. Esto abrirá una lista de hosts disponibles. Aquí es donde debes agregar un nuevo host SSH. Para hacerlo, selecciona la opción "Add New SSH Host".



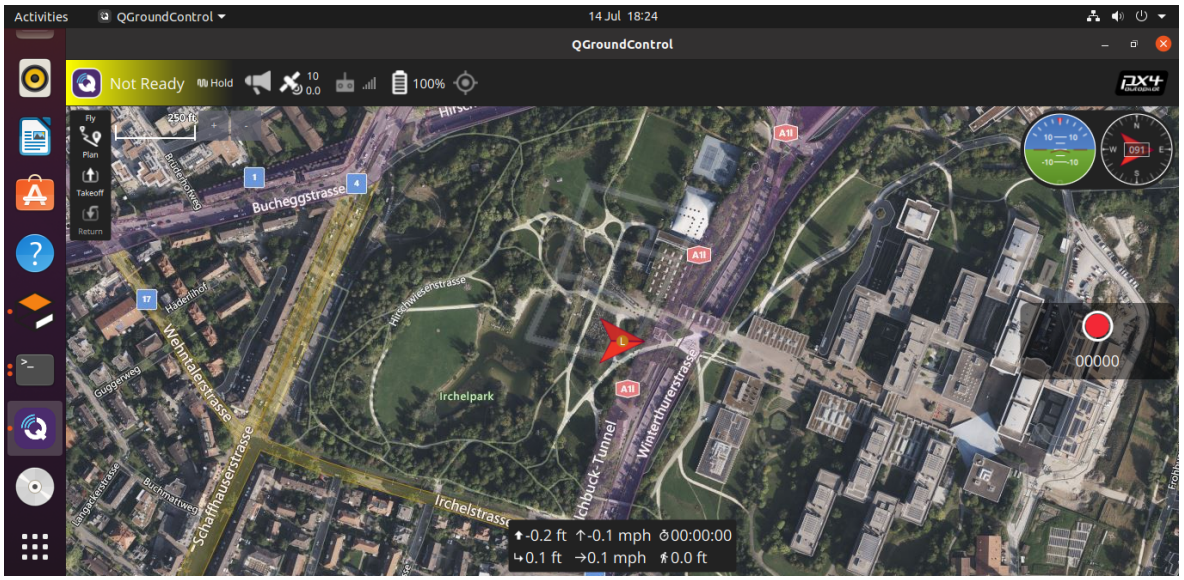
Add New SSH Host

4. En la ventana emergente, ingresa la información de conexión en el formato "ssh user@host". Asegúrate de reemplazar "user" por tu nombre de usuario y "host" por la dirección IP o nombre de dominio de tu Jetson Nano.
5. Después de agregar el nuevo host SSH, regresa a la lista de hosts disponibles y selecciona el host que acabas de agregar.
6. En este punto, se te solicitará ingresar la contraseña asociada con el usuario SSH en tu Jetson Nano.
7. Una vez que hayas proporcionado la contraseña correcta, la conexión SSH se establecerá y podrás comenzar a configurar la Jetson Nano a través de la línea de comandos.

Anexo 2 - Como hacer misión en QgroundControl

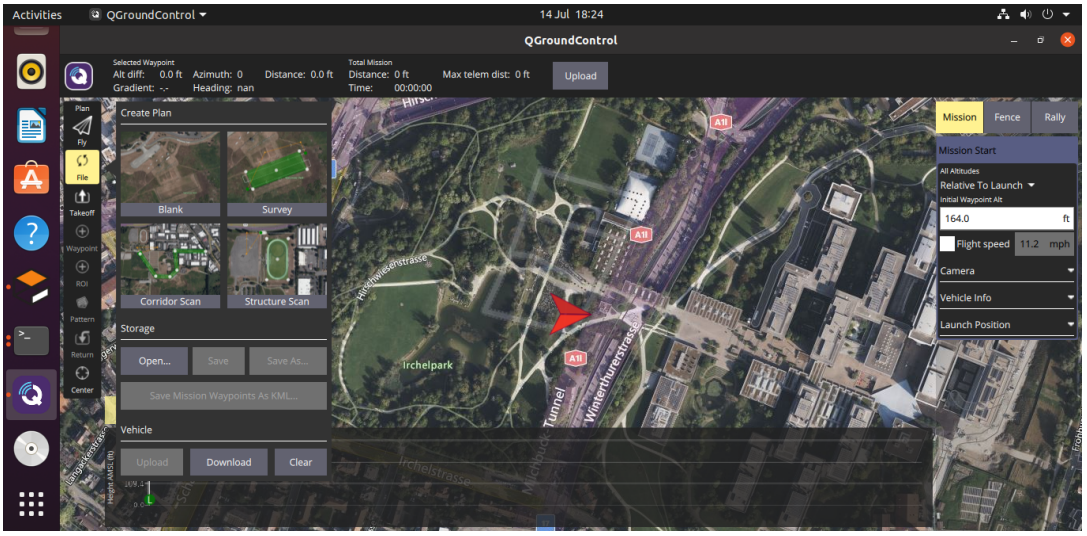
Para hacer una misión en qground control es bastante sencillo, se pueden seguir los siguientes pasos :

1. Abrir la aplicación QGroundControl.
2. En la parte izquierda de la interfaz, buscar y seleccionar la pestaña "Plan" o "Planificación".



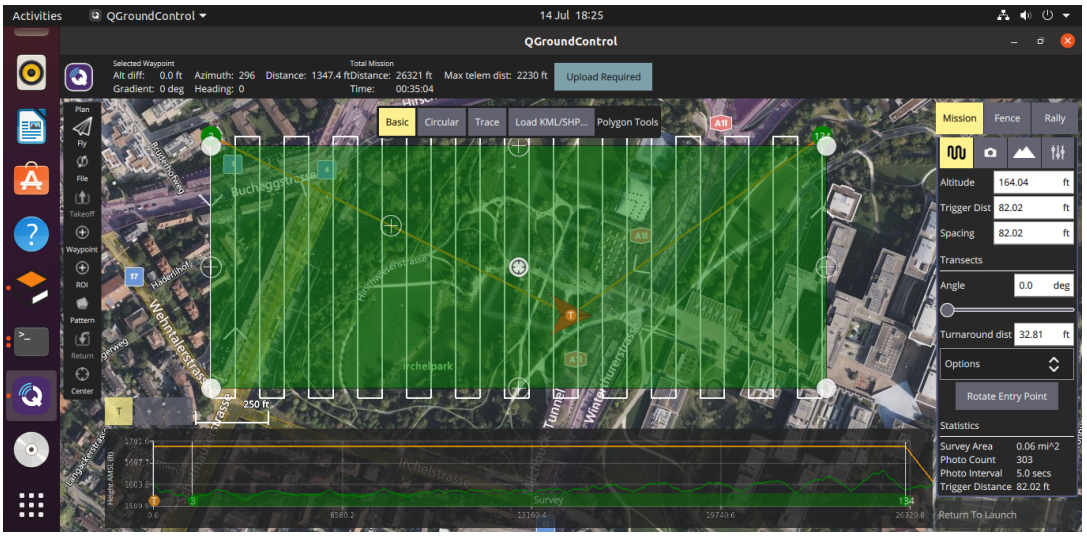
QGroundControl

3. A continuación, elegir el tipo de patrón de vuelo que se desea utilizar. En este caso, seleccionar la opción "Rectangular" o "Rectángulo".



QGroundControl escoger misión

4. Al seleccionar el patrón rectangular, se abrirá una vista en el área de trabajo donde se podrá definir los puntos de la misión.

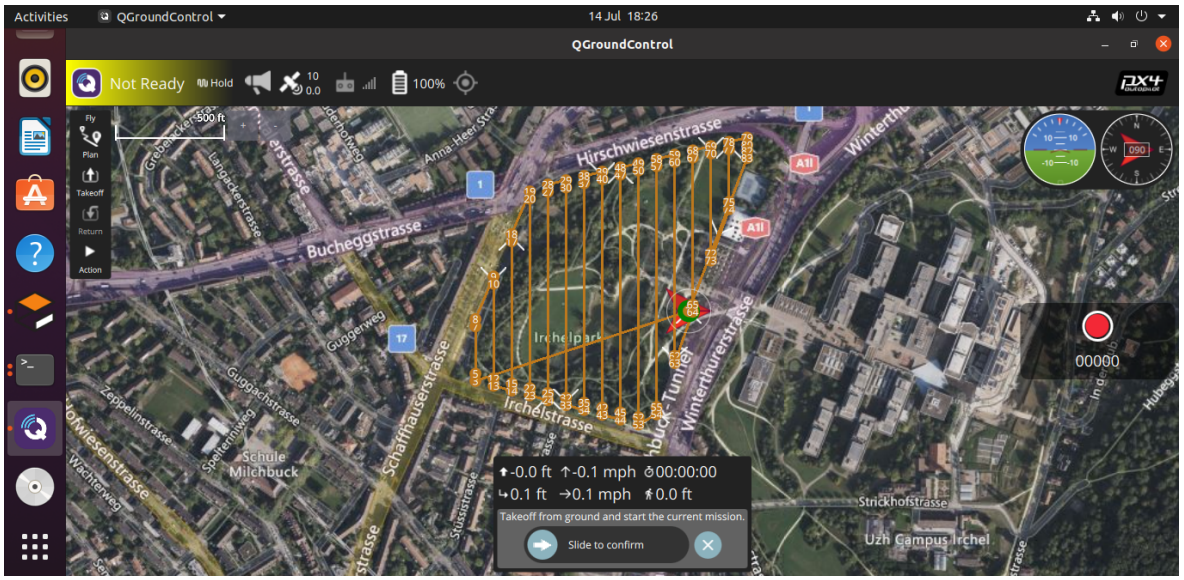


QGroundControl ajustar misión

5. Hacer clic en el mapa para establecer el primer punto de la misión. Luego, hacer clic en otros puntos para definir la forma y el tamaño del rectángulo de vuelo. El último punto que se seleccione será el punto de inicio y finalización de la misión.

6. Una vez que se haya establecido los puntos de la misión, se pueden ajustar otros parámetros como la altitud de vuelo, velocidad, orientación, entre otros. Estos ajustes dependerán de las necesidades específicas de la misión.
7. Revisar cuidadosamente todos los ajustes y parámetros de la misión para asegurarte de que estén configurados correctamente.
8. Guardar la misión o hacer clic en "Enviar a Vehículo" para transferirla al dron o vehículo aéreo no tripulado (UAV). Asegurarse de que el dron esté conectado a QGroundControl para que la misión se pueda transferir correctamente.

Una vez que se hayan completado estos pasos, la misión rectangular estará lista para ser ejecutada en el dron. Recordar siempre seguir las regulaciones locales y las mejores prácticas de seguridad al realizar misiones de vuelo con drones.



Misión lista para comenzar

Anexo 3 -Instalar git

1. Instalar git una herramienta de control de versiones, mediante el siguiente comando:

```
Unset
sudo apt-get install git
```

Configurar Git para establecer la información de usuario, como el nombre y el correo electrónico. Esto es importante para realizar un seguimiento adecuado de los cambios y contribuciones. Puedes configurarlo utilizando los siguientes comandos:

```
Unset
git config --global user.name "Tu Nombre"
git config --global user.email "tu@email.com"
```

Se debe reemplazar "Tu Nombre" y "tu@email.com" con la información personal.