* **Reporte del Avance – Proyecto de Servicio Social -**

**Realizado por:** Mauro Ortiz Juárez (La Salle 2do semestre Bachillerato)

**Titulares:** Dra. Cosío y Dra. Anabel

**Periodo de trabajo (Mauro):** 10/6/25 – 24/6/25

**Universidad Politécnica de Pachuca**

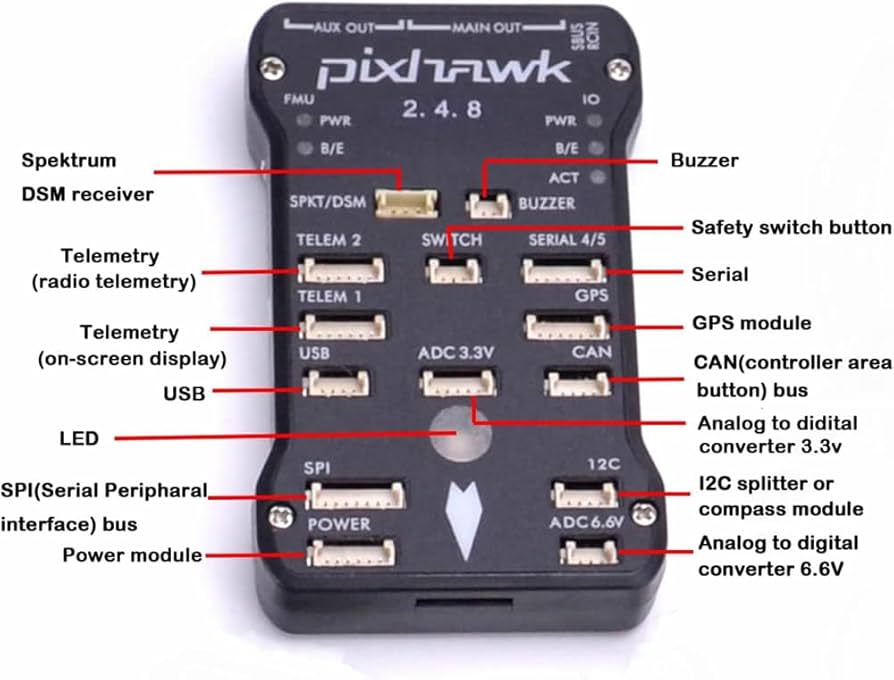
* **Objetivo del Proyecto** –

Crear un sistema autónomo basado en drones que permita analizar de forma automática zonas afectadas por desastres naturales, utilizando visión por computadora (Librería OpenCV), Machine Learning de YOLOv5, Algoritmos, y comunicación a distancia, con el propósito de dar comunicaciones en zonas de difícil o nulo acceso, así como apoyar en tareas como la búsqueda de víctimas, localizar teléfonos celulares para ubicar a las personas afectadas y brindar información de las áreas/personas afectadas.

TODOS LOS ARCHIVOS NECESARIOS (ASÍ COMO UN VIDEO) SE ENCUENTRAN EN ESTA MEMORIA USB

**Introducción al proyecto:**

* **Componentes con su función y características**
* **Librerías necesarias**
* **Comunicaciones utilizadas**
* **Descripción estructural del sistema completo**
  + - 1. **Componentes con su función y características**



1. **Pixhawk 2.4.8 (Controladora de vuelo)**

* Se encarga de la estabilización y navegación del dron
* Soporta el firmware de ArduPilot
* Tiene entradas para GPS, sensores, y módulos de telemetría

1. **GPS M8N**

* Proporciona posicionamiento mediante satélites GNSS
* Permite que el dron vaya a coordenadas específicas de manera autónoma
* Da una posición GEO en una imagen en vivo vía satélite

1. **Nvidia Jetson Nano**

* Minicomputadora con arquitectura ARM y GPU integrada.
* Corre el modelo de visión por computadora (OpenCV +YOLOv5) para detectar escenarios
* Corre Python y puede comunicarse con Pixhawk usando MAVLink

1. **3DR Sik Telemetry Radio 915 MHz**

* Sistema de comunicación entre el dron y la estación base
* Transmite telemetría en tiempo real: ubicación, estado del dron, comandos, etc
* Funciona por puerto serial y permite distancias de varios kilómetros



1. **Cámara Raspberry HQ (en Jetson)**

* Proporciona la imagen para el modelo de detección YOLOv5
* Brinda una excelente calidad de Imagen – requiere un lente adicional

1. **Batería LiPo 4200 mAh**

* Proporciona energía al dron
* Voltaje y capacidad adecuados para unos minutos de vuelo estable con carga
  + - 1. **Librerías Necesarias**

1. **PyTorch**

* Framework para Deep Learning necesario para correr YOLOv5

1. **OpenCV**

* Permite la captura y procesamiento de imágenes desde la cámara

1. **YOLOv5 (Ultralytics)**

* Red neuronal pre-entrenada para detección de objetos en tiempo real
* Permite identificar personas y contar cuántas hay en pantalla

1. **pymavlink**

* Permite enviar comandos al Pixhawk desde Jetson usando el protocolo MAVLink

1. **NumPy**

* Manipulación rápida de datos numéricos e imágenes

1. **serial / pyserial**

* Para la comunicación entre Jetson y Pixhawk por puerto serial (UART)

**DIAGRAMA DE FLUJO DE COMUNICACIÓN**

* **Pixhawk** ejecuta la misión cargada (por ejemplo, ir a coordenadas específicas)
* **Jetson Nano**, conectada por UART a la Pixhawk, va analizando video en tiempo real con **YOLOv5**.
* Dependiendo el script, **Jetson Nano** enviará comandos para que el **dron** los realice
* Toda la telemetría de vuelo (posición, altura, batería, etc.) se transmite desde Pixhawk a la **estación base** mediante los **módulos 3DR Sik**
* La estación base puede enviar nuevos comandos o simplemente monitorear el estado en **Mission Planner**

**[ Estación base]**

**▲**

**│ (915 MHz - 3DR Sik + Antena Yagi)**

**▼**

**[ Dron en vuelo]**

**┌───────────────────────┐**

**│ Pixhawk 2.4.8 │◄───── GPS M8N**

│ - Recibe comandos │

│ - Controla motores │

│ - Ejecuta misión **│**

**└────────▲──────────────┘**

**│ (UART - MAVLink) (Micro USB – USB -A)**

**▼**

**┌───────────────────────┐**

**│ Jetson Nano │**

**│** - Procesa video (YOLOv5) **│**

**│** - Detecta personas **│**

**│** - Toma decisiones **│**

**└────────▲──────────────┘**

**│**

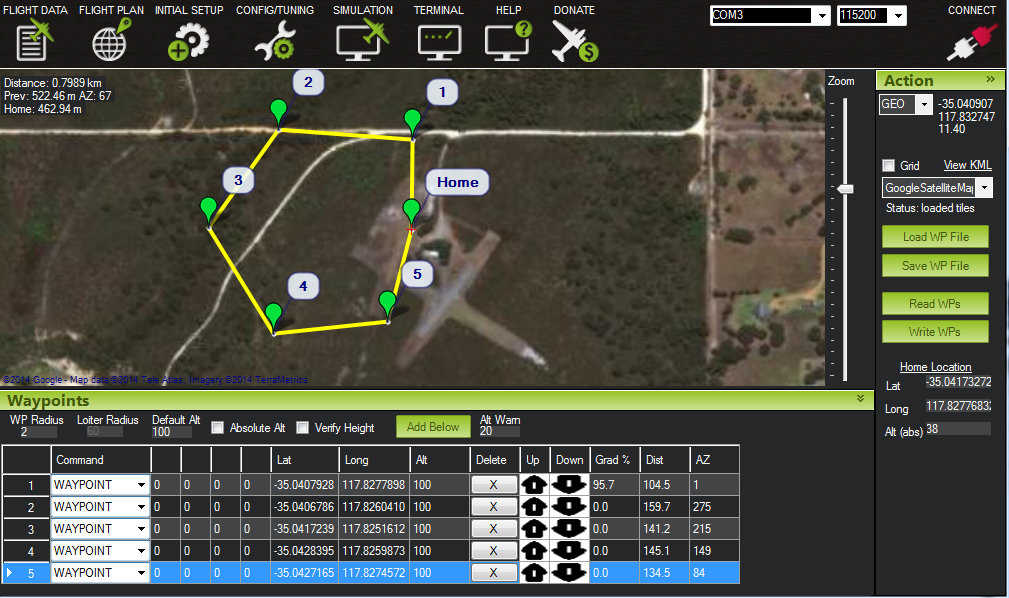
**Cámara**

**ACERCA DE MISSION PLANNER –**

Mission Planner es un software de control y configuración Open-source (C#) de drones que utilizan el firmware ArduPilot (como la Pixhawk y PX4). Es una herramienta integral que permite desde la planificación de misiones autónomas hasta la calibración de sensores, el monitoreo en tiempo real del dron y el análisis del vuelo.

Está diseñado principalmente para usarse en PC con Windows, y se conecta al dron por USB, telemetría (3DR Sik 915 MHz) o incluso WiFi (WiFi contiene limitaciones de distancia y obstáculos). Al conectar la Pixhawk al programa, ya sea por medio de USB o mediante módulos de telemetría como los 3DR Sik, se puede observar en tiempo real toda la información de vuelo: ubicación GPS, altitud, orientación, nivel de batería, estado de los sensores etc.

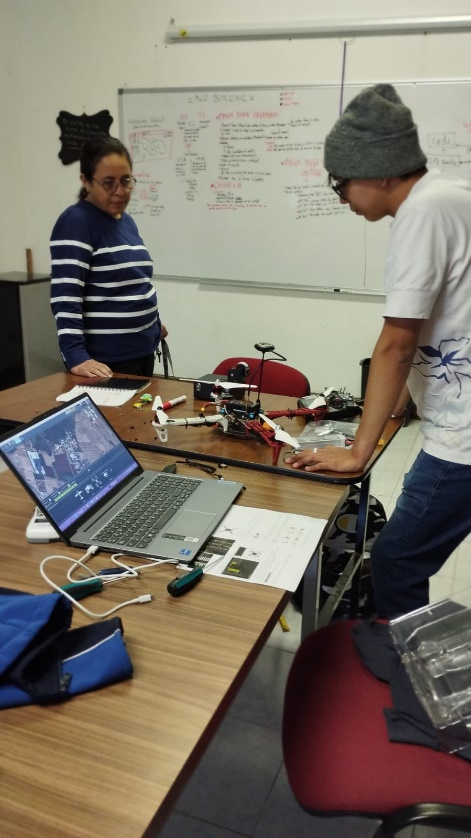
Una de las características más importantes de Mission Planner es la posibilidad de crear misiones autónomas. Esto nos dice que el usuario puede trazar en un mapa digital los puntos por donde quiere que el dron vuele, definiendo coordenadas, altitudes, velocidades y acciones como detenerse, esperar o incluso tomar fotografías.

En este caso, este software se está utilizando para hacer que el dron sea completamente autónomo y actúe en base a la programación y algoritmos que este tenga. Mission Planner nos ayuda a establecer funciones como RTL (Return To launch) o LOTIER (Orbitar o dar vueltas en una dirección). Estas funciones se encuentran definidas en Arupilot en su repositorio oficial (<https://github.com/ArduPilot/ardupilot/tree/master/ArduCopter>). De esta manera nos ayuda a establecerlo en el código de Python (en el archivo CODIGOPRUEBA (TEST) podemos ver un ejemplo aplicado, en el que el código se basa en la detección de 5 o más personas (función/invocar modo deseado: vehicle.mode = VehicleMode("RTL") ).

Interfaz principal de Mission Planner



Mapeo de misión a ejecutar en el Dron

**EVIDENCIAS FOTOGRÁFICAS**

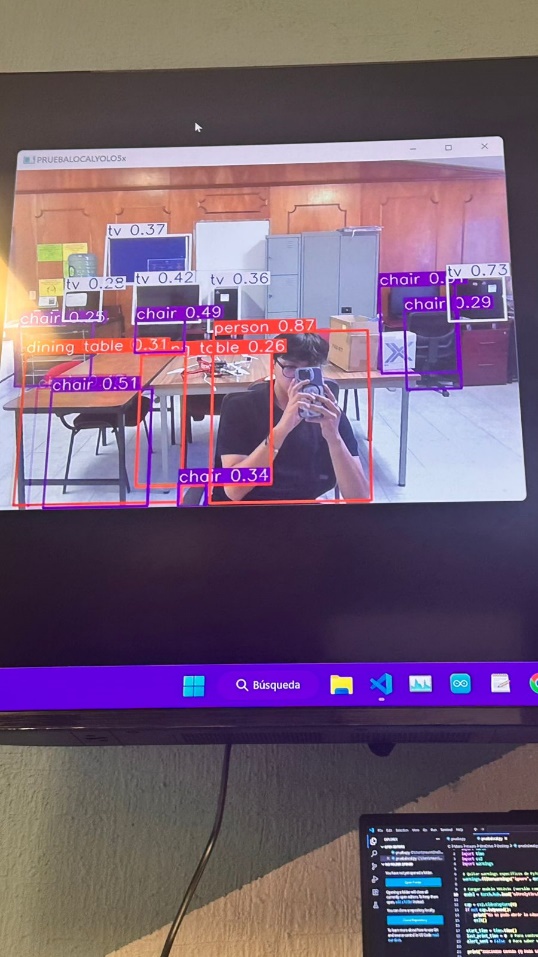
Una pantalla de televisión encendida

El contenido generado por IA puede ser incorrecto.



Interfaz de usuario gráfica

El contenido generado por IA puede ser incorrecto.Imagen que contiene interior, tabla, pequeño, puesto

El contenido generado por IA puede ser incorrecto.

Texto, Pizarra

El contenido generado por IA puede ser incorrecto.

Texto, Pizarra

El contenido generado por IA puede ser incorrecto.

**Actividades Realizadas** –

En la siguiente tabla se mostrarán las actividades realizadas a lo largo del periodo

10/6/25 – 24/7/25

|  |  |
| --- | --- |

| **Fecha** | **Actividad** |
| --- | --- |
| 10 junio | Presentación del proyecto en general, algoritmo genético, y visualización del hardware + software disponible:  -Kit HAWK'S WORK F450 Drone Kit, Marco + Pixhawk 4.9.8+ GPS M8N+ Módulo de Potencia + ESC + Motores sin Escobillas + Hélices + Batería 42200 mAh Lipo + Transmisor y Receptor RC 2.4Ghz + Ardupilot |
| 12 junio | Instalación y configuración inicial del Software Mission Planner, al igual que se llevó acabo el ensamblado (parcial) del dron junto con la planificación de el funcionamiento de la base y la comunicación con este mismo |
| 17 junio | Se llevó acabo la configuración del barómetro del dron + calibración del módulo GPS M8N para geolocalizar el dron en el software Mission Planner + carga de firmware de Ardupilot en placa Pixhawk |
| 19 junio | Corrección del ensamblaje del dron entre los motores y tornillos (los motores se encontraban al revés, y los tornillos mal ajustados). Se realizó por primera vez el vuelo (test) del dron en un área controlada + calibración de PID y estabilización |
| 24 junio | Creación del diagrama de las conexiones entre los módulos de comunicación del dron y base (al final decidimos utilizar estos módulos de radio = 3DR SIK Radio Telemetry Kit), así como la investigación de librerías, Machine Learning y protocolos necesarios en Python para realizar el código e implementarlo en una tarjeta Nvidia Jetson (Nano) |
| 26 junio | Descarga e instalación de librerías YOLOv5, Opencv, Cv2, git, pip, pytorch (torch), Dronekit, Python 3.10 y pymavlink en computadora Windows. Se realizaron pruebas de compatibilidad y descarga de dependencias, así como librerías necesarias en el entorno VSCode para llevar a cabo un script para reconocer el entorno con visión por computadora |
| 1 julio | Instalación de Ubuntu + Jetpack en la placa Nvidia Jetson (Nano), al igual que dependencias necesarias para el funcionamiento futuro del código |
| 3 julio | Desarrollo de script de simulación en VSCode con una lógica de detección de personas (>=5 personas detectadas, el dron realiza un reconocimiento de la zona). Diseño del flujo de misión autónoma: despegue, navegación, detección y retorno + Compra de Módulos 3DR SIK Radio Telemetry Kit para la comunicación del dron a base. |
| 10 julio | Se comenzaron a realizar simulaciones en el software Mission Planner con SITL (Software In The Loop), en el cual se probaron comandos de: WPs, (Waypoints / puntos de ida) TAKE\_OFF, (Despegar dron(es)) RTL, (Return To Launch) y visualización de telemetría (simulación) del dron en tiempo real. Se descargó Ubuntu 20.04 en una SD, lista para ser flasheada en la Jetson Nano. |

**Logros Alcanzados -**

* Se configuró correctamente la Pixhawk y el GPS M8N
* Se ensambló por completo el dron
* Pruebas de vuelo reales (manuales) con el dron
* Se instaló parcialmente YOLOv5 en la Jetson Nano
* Se diseñó el script de misión autónoma basado en detección de personas
* Se realizaron pruebas de simulación en consola de Mission Planner que nos da el comportamiento esperado (al detectar >5 personas con OpenCV, el dron manda STATUS\_TEXT a la consola)
* Se realizó la compra de los módulos de radiofrecuencia 3DR SIK RADIO 915 MHz

**Problemas encontrados -**

* La tarjeta Nvidia Jetson (Nano) no se actualiza su software (Ubuntu 17 – Ubuntu 20.0.4)
* El dron tiene pequeñas oscilaciones en la estabilidad
* En la Nvidia Jetson (Nano), la librería Pytorch requiere otros elementos que deben ser instalados externa o manualmente (depende la versión de Python, Ubuntu, y Jetpack)
* El procesamiento de YOLOv5 en la Nvidia Jetson (Nano) va a depender del modelo que utilicemos (>Modelo = mayor tiempo de respuesta + más procesamiento de imágenes --- <Modelo = menos tiempo de respuesta + menos procesamiento de imágenes)
* Métodos limitados para la localización de los celulares de las víctimas

**Siguientes pasos -**

* Completar la instalación completa de YOLOv5 + librerías con la cámara Raspberry HQ montada en la Nvidia Jetson (Nano)
* Hacer pruebas físicas y reales de vuelo en entorno controlado de manera autónoma
* Desarrollar comunicación funcional entre la Nvidia Jetson (Nano) y Pixhawk mediante el protocolo MAVLink + Módulos 3DR SIK 915 MHz
* Integrar el algoritmo genético al código de la Nvidia Jetson (Nano)
* Realizar estimaciones de datos de vuelo + corrección de errores al realizar las pruebas

|  |  |
| --- | --- |