

Identificación “in situ” de grafemas de la  
escritura maya en monumentos arqueológicos  
empleando Inteligencia Artificial.  
Reporte de Servicio Social

Septiembre 9, 2020

Fecha de inicio: Septiembre 1, 2019  
Alumno: Juan Daniel Soto Montes  
Asesor: J. Guadalupe Pérez Ramírez

## 1. Objetivo

Desarrollar un drone con capacidad de acercarse, de manera autónoma, a un monumento que contenga escritura maya en una zona arqueológica, e identificar los grafemas en la escritura maya.

### 1.1. Abreviaciones

**NN** Red Neuronal Artificial.

**CNN** Red Neuronal Convolucional.

## 2. Material

Se adquirió un drone de la marca “Eachine” modelo “Wizard X220S” para realizar un vuelo autónomo.



Figura 1: Drone Eachine Wizzard220s

Para controlar el movimiento del drone, se adquirió tambien un radio-control compatible con el drone.



Figura 2: Radio Control FLYSKY

Otro aspecto importante para operar el drone es la recepción del video en tiempo real de la cámara que incorpora el drone y que se puede observar en una computadora como entrada de video.



Figura 3: Antena Receptora de Video

Una microcomputadora JetsonNano se encargara de realizar diversas funciones que le permitiran al drone volar de manera autónoma.

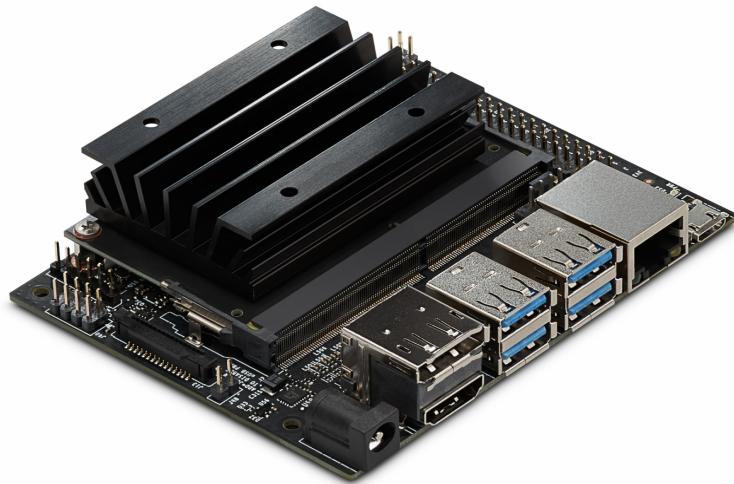


Figura 4: Microcomputadora NVIDIA JetsonNano

### **3. Actividades**

Durante el desarrollo del servicio social se llevaron a cabo diversas actividades, las cuales serán descritas dentro de esta sección. Cada una de estas se realizó con la intención de cumplir con el objetivo principal.

#### **3.1. JetsonNano**

La micro computadora jetson nano es un pequeño ordenador, de gran alcance que le permite ejecutar múltiples redes neuronales en paralelo para aplicaciones como la clasificación de imágenes, detección de objetos, segmentación y procesamiento del habla.

Las características de esta computadora son:

GPU	Maxwell de 128 núcleos
UPC	BRAZO de cuatro núcleos A57 @ 1.43 GHz
Memoria	LPDDR4 de 4 GB y 64 bits 25.6 GB/s
Almacenamiento	microSD (no incluida) Se recomienda: 64gb U1xx
Codificación de video	4K @ 30 — 4x 1080p @ 30 — 9x 720p @ 30 (H.264 / H.265)
Decodificación de video	4K @ 60 — 2x 4K @ 30 — 8x 1080p @ 30 — 18x 720p @ 30 (H.264 / H.265)
Cámara	1x carriles MIPI CSI-2 DPHY
Conectividad	Gigabit Ethernet, M.2 Clave E
Monitor	HDMI 2.0 y eDP 1.4
USB	4x USB 3.0, USB 2.0 Micro-B
Otros	GPIO, I 2 C, I 2 S, SPI, UART

Cuadro 1: Carácteristicas de Microcomputadora JetsonNano.

##### **3.1.1. Hardware**

Hardware necesario para empezar a usar la tarjeta *JetsonNano*:

- Fuente de poder 5vDC 3.5A.
- Monitor con entrada HDMI.
- Tecaldo y mouse.
- Antena wi-fi o conexión con cable ethernet.
- Memoria microSD de almenos 64Gb u1 (velocidad de escritura de 10MB/s).
- Adaptador para conectar la memoria a una computadora.

### 3.1.2. Instalar Sistema Operativo

La microcomputadora *JetsonNano* requiere de una memoria externa microSD donde se instala su sistema operativo, el cual es proporcionado por NVIDIA. La imagen que se usará para montar en la memoria se encuentra en:

<https://developer.nvidia.com/jetson-nano-sd-card-image-r322>

Este proceso se realizó usando una computadora con sistema operativo Ubuntu 16.04. En Ubuntu se puede usar la herramienta disks para formatear la tarjeta sd:

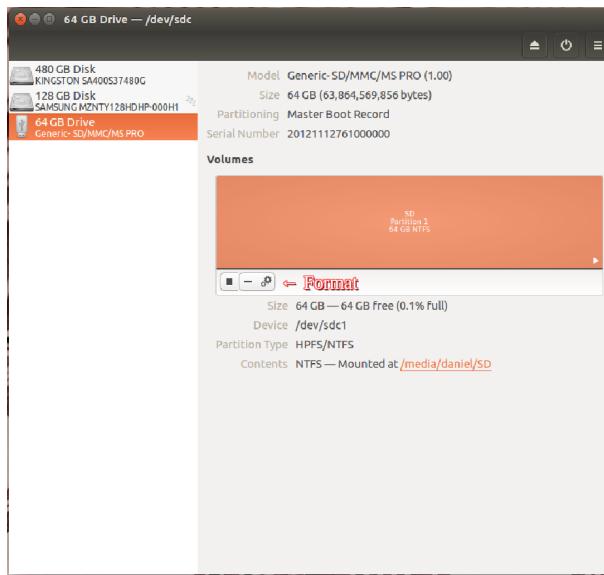


Figura 5: Herramienta Disk

Una vez formateada la memoria, se debe de eliminar el volumen haciendo click en “ - ”, una vez eliminado el volumen, se debe de ver de la siguiente manera:

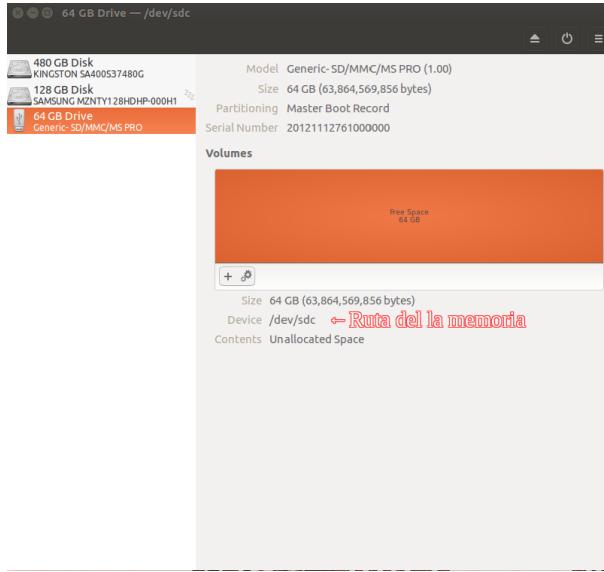


Figura 6: Dispositivo formateado y desmontado

Para grabar la imagen sobre la tarjeta SD, se abre una terminal (ctrl + alt + t) y nos dirigimos a la ruta donde se encuentra la imagen descomprimida y comprobamos la ruta de la memoria con el comando **lsblk**, la ruta debe coincidir con la vista dentro de la herramineta disk:

```
daniel@daniel:~/Downloads/Jetson$ ll
total 17867312
drwxrwxr-x 2 daniel daniel 4096 Aug 16 18:19 /
drwxrwxrwx 27 daniel daniel 4096 Aug 16 18:19 [REDACTED]
-rw-rw-r-- 1 daniel daniel 5411266762 Aug 15 17:57 jetsonNano.zip
-rw-r--r-- 1 daniel daniel 12884901888 Jul 16 22:58 sd-blob-b01.img
daniel@daniel:~/Downloads/Jetson$ lsblk
NAME MAJ:MIN RM SIZE RO TYPE MOUNTPOINT
sd[0-9] 8:16 0 119.2G 0 disk
|__sdb[0-9] 8:20 0 482M 0 part
|__sdb2 8:20 0 100M 0 part
|__sdb3 8:21 0 380M 0 part
|__sdb3 8:19 0 117.5G 0 part /media/daniel/SSD
|__sdb1 8:17 0 260M 0 part /boot/efi
sdc 8:32 1 59.5G 0 disk - Esta es la memoria
sda 8:0 0 447.1G 0 disk
└─sda1 8:1 0 447.1G 0 part /
daniel@daniel:~/Downloads/Jetson$
```

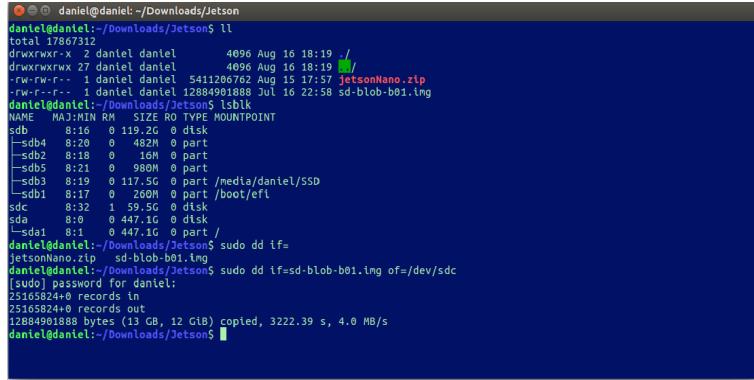
Figura 7: Identificación de dispositivo en la terminal

Para montar la imagen en la memoria se usa el comando **dd**:

```
$ sudo dd if=/ruta-de-iso of=/ruta-de-la-memoria
```

Este comando puede tardar algunos minutos, no hace ninguna indicación de su avance, por lo que para monitorear si sigue activo se puede usar el comando **top**

en una terminal diferente. Cuando se termine de grabar la imagen, la terminal deberia verse de la siguiente manera:



```
daniel@daniel: ~/Downloads/Jetson$ ll
total 17867312
drwxrwxr-x  2 daniel daniel    4096 Aug 16 18:19 /
drwxrwxr-x 27 daniel daniel    4096 Aug 16 18:19 JetsonNano.zip
-rw-r--r--  1 daniel daniel 5411286762 Aug 15 17:57 JetsonNano.zip
-rw-r--r--  1 daniel daniel 12884901888 Jul 16 22:58 sd-blob-b01.img
daniel@daniel:~/Downloads/Jetson$ lsblk
NAME MAJ:MIN RM SIZE RO TYPE MOUNTPOINT
sdb  8:16   0 119.2G  0 disk
└─sdb4 8:20   0 482M  0 part
  └─sdb2 8:18   0 16M  0 part
  └─sdb5 8:21   0 980M  0 part
  └─sdb3 8:19   0 117.5G 0 part /media/daniel/SSD
  └─sdb1 8:17   1 260M  0 part /boot/efl
sdc  8:32   1 536.9G 0 disk
└─sdc1 8:33   0 447.1G 0 part
  └─sda1 8:1   0 447.1G 0 part /
daniel@daniel:~/Downloads/Jetson$ sudo dd if=
jetsonNano.zip of=sd-blob-b01.img
daniel@daniel:~/Downloads/Jetson$ sudo dd if=sd-blob-b01.img of=/dev/sdc
[sudo] password for daniel:
25165824+0 records in
25165824+0 records out
12884901888 bytes (13 GB, 12 GiB) copied, 3222.39 s, 4.0 MB/s
daniel@daniel:~/Downloads/Jetson$
```

Figura 8: Imagen ISO grabada

Una vez grabada la memoria con el sistema operativo, solo falta insertar la tarjeta SD donde corresponde en la tarjeta:



Figura 9: Insercion de la tarjeta microSD grabada

Una vez colocada la memoria, se deberá conectar el monitor, el cable ethernet o antena wifi (opcional), el teclado, el mouse y la fuente, una vez conectada la fuente, la tarjeta se encenderá automáticamente, no se debe hacer nada hasta que en el monitor aparezca lo siguiente:

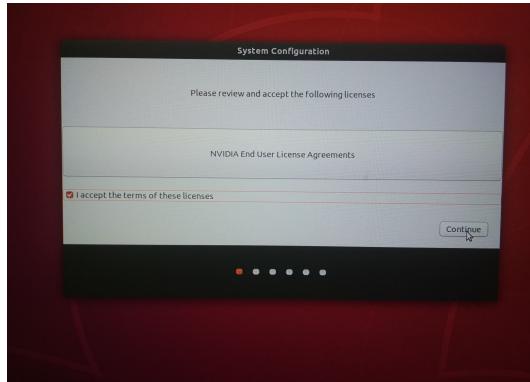


Figura 10: Menu de Instalación de Sistema Operativo

A continuación se presentara la configuración de inicio, esta se deberá seguir y dependiendo la preferencia o necesidad. Será necesario establecer un nombre de usuario, así como una contraseña de administrador que servira para realizar operaciones de super usuario, una vez terminada esta configuración la tarjeta estará lista para usarse.



Figura 11: Sistema Operativo Instalado

El sistema operativo instalado es Ubuntu 18.04, en una versión mínima proporcionada por NVIDIA especialmente para sus microcomputadoras de desarrollo Jetson.

### 3.1.3. Pines GPIO

La tarjeta incluye también 6 unidades de memoria, de las cuales solo una se encuentra montada: L4T-README se encuentra visible para lectura y escritura, dentro de la partición se encontraron 5 archivos de texto:

**INDEX.txt** Describe el contenido de la partición.

**l4t-serial.inf** información del dispositivo.

**README-usb-dev-mode.txt** Explica cómo linux puede guardar archivos en usb así como demás periféricos dev para Tegra.

**README-vnc.txt** Indica cómo ejecutar VNC (escritorio remoto).

**README-wifi** Explica cómo conectarse con la tarjeta jetson mediante ssh y cómo conectar la tarjeta a internet wifi.

Estos archivos son solo descriptivos, ya que su contenido se encuentra preinstalado.

En el escritorio de la computadora aparecen 3 iconos de nvidia que son accesos directos a foros de nvidia, así como una wiki: [https://elinux.org/jetson\\_Zoo](https://elinux.org/jetson_Zoo) esta wiki contiene instrucciones para instalar paquetes de open-source y frameworks de Nvidia-Jetson, así como una colección de modelos de inferencia.

En la pagina <https://www.jetsonhacks.com/2019/06/07/jetson-nano-gpio/> nos dice que para tener acceso a los puertos GPIO debemos abrir una terminal y entrar como superusuario usando el comando **sudo su** y después agregando los permisos al usuario que lo usará escribiendo lo siguiente en la terminal:

```
$ sudo groupadd -f -r gpio  
$ sudo usermod -a -G gpio usuario
```

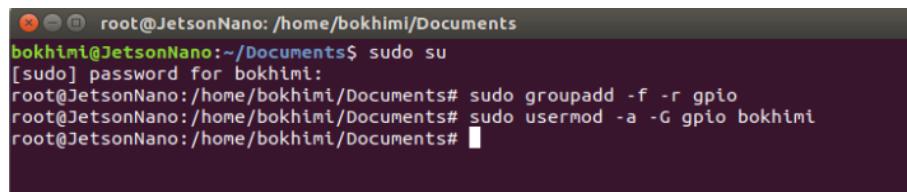
A screenshot of a terminal window titled "root@JetsonNano: /home/bokhimi/Documents". The window shows the command "sudo su" being run, followed by the password entry for "bokhimi". Then, the commands "sudo groupadd -f -r gpio" and "sudo usermod -a -G gpio bokhimi" are executed successfully. The terminal prompt "root@JetsonNano:/home/bokhimi/Documents#" is visible at the bottom.

Figura 12: Otorgando permisos para usar pines de GPIO

Luego actualizamos las reglas para usar los dispositivos dev con:

```
$ sudo cp /opt/nvidia/jetson-gpio/etc/99-gpio.rules /etc/udev/rules.d/  
$ sudo udevadm control -reload-rules && sudo udevadm trigger
```

Ahora, para usar uno de los ejemplos, en la ruta /opt/nvidia/jetson-gpio/samples ejecutamos el ejemplo:

```
$ cd /opt/nvidia/jetson-gpio/samples  
$ sudo ./run_sample.sh simple_out.py
```

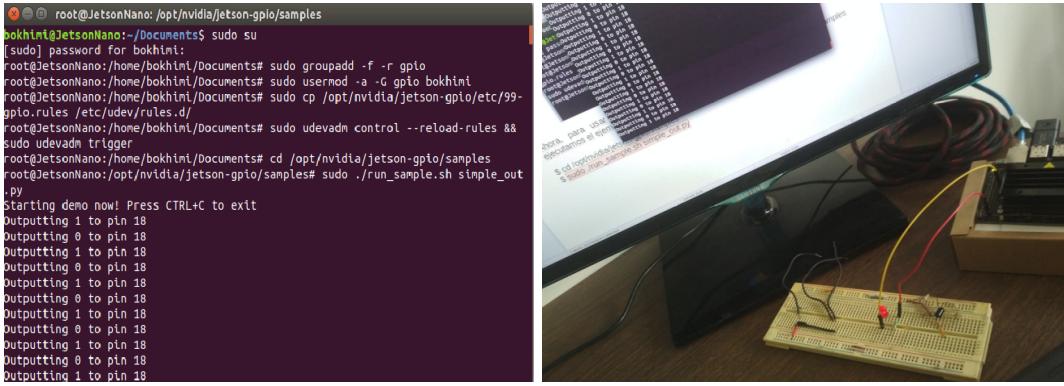


Figura 13: Ejemplo de uso de los pines GPIO

Se instaló el editor de textos nano para poder editar el programa desde la terminal con:

```
$ sudo apt-get install nano
```

Analizando el programa, se encontró que el bash es un arrancador para los pines de la tarjeta, y el programa que controla los puertos esta escrito en python y necesita de 2 bibliotecas “RPi.GPIO” y “time”, donde la primera es para declarar los pines y time es un contador de la tasa de reloj de la computadora.

Los pines GPIO de la tarjeta Jetson Nano están ordenados de la siguiente manera:

Jetson Nano J41 Header							<a href="http://www.neko.ne.jp/~freening/">http://www.neko.ne.jp/~freening/</a>	
Pi GPIO#	Sysfs GPIO	Name	Pin	Pin	Name	Sysfs GPIO	Pi GPIO#	
		3.3VDC Power	1	2	5.0VDC Power			
2		SDA1 I2C Bus 1	3	4	5.0VDC Power			
3		SCL1 I2C Bus 1	5	6	GND			
4	gpio216	AUDIO_MCLK	7	8	TXDO		14	
		GND	9	10	RXD0		15	
17	gpio50	UART2_RTS	11	12	DAP4_SCLK	gpio79	18	
27	gpio14	SPI2_SCK	13	14	GND			
22	gpio194	LCD_TE	15	16	SPI2_CS1	gpio232	23	
		3.3VDC Power	17	18	SPI2_CS0	gpio15	24	
10	gpio16	SPI_MOSI	19	20	GND			
9	gpio17	SPI_MISO	21	22	SPI2_MISO	gpio13	25	
11	gpio18	SPI1_SCK	23	24	SPI1_CS0	gpio19	8	
		GND	25	26	SPI1_CS1	gpio20	7	
(0)		ID_SDA I2C Bus 0	27	28	ID_SCL I2C Bus 0		(1)	
5	gpio149	CAM_AF_EN	29	30	GND			
6	gpio200	GPIO_P20	31	32	LCD_BL_PWM	gpio168	12	
13	gpio38	GPIO_PE6	33	34	GND			
19	gpio76	DAP4_FS	35	36	UART2_CTS	gpio51	16	
25	gpio12	SPI2_MOSI	37	38	DAP4_DIN	gpio77	20	
		GND	39	40	DAP4_DOUT	gpio78	21	

JetsonHacks  
<https://www.jetsonhacks.com/nvidia-jetson-nano-j41-header-pinout/>

Figura 14: Etiquetas de los pines GPIO

En esta tabla se puede apreciar que la forma en cómo se declaran los pines en el programa, no es exactamente igual a las etiquetas que tiene la tarjeta para numerar los pines, esta consideración debe hacerse para que se pueda usar de manera correcta los pines.

En la página [https://elinux.org/Jetson\\_Zoo#Hello\\_AI\\_World](https://elinux.org/Jetson_Zoo#Hello_AI_World) indica como descargar una carpeta de ejemplos de inferencia con redes ya entrenadas y listas para usar, y se puede instalar de la siguiente manera:

– Download

```
$ git clone https://github.com/dusty-nv/jetson-inference  
$ cd jetson-inference  
$ git submodule update --init
```

– Configure build tree

```
$ mkdir build  
$ cd build  
$ cmake ..
```

– Build and install

```
$ make  
$ sudo make install
```

Como se requiere la instalacion de la biblioteca de Python Jetson.GPIO en Jetpack 4.2, para poder trabajar con los pines de GPIO, para lo cual se intaló pip con:

– Setup groups/permissions

```
$ sudo groupadd -f -r gpio  
$ sudo usermod -a -G gpio bokhimi  
$ sudo cp /opt/nvidia/jetson-gpio/etc/99-gpio.rules /etc/udev/rules.d/  
$ sudo udevadm control -reload-rules && sudo udevadm trigger
```

– Reboot required for changes to take effect

```
$ sudo reboot  
$ sudo apt update  
$ sudo apt install python3-pip  
$ sudo pip3 install Jetson.GPIO
```

Esta biblioteca remplazó en el código a la biobliboteca Rpi.GPIO que es la biblioteca que usa Raspberry py, y con la boblioteca Jetson.GPIO esta hecha para trabajar con los puertos de las tarjetas Jetson.

También se cambió el compilador por defecto de python de la versión 2.7.15+, a la versión 3.6.8 usando los siguientes comandos:

```
$ update-alternatives --install /usr/bin/python python /usr/bin/python2.7 1
$ update-alternatives --install /usr/bin/python python /usr/bin/python3.6 2 ,
$ python --version
```

Esto con el fin de usar herramientas que usar este compilador posteriormente.

### **3.2. Drone**

En esta actividad se plantó conocer el hardware del drone, así como el programa que usa para poder volar, su comunicación para el control por radiocontrol y su recepción de video. Tomando en cuenta que el drone adquirido es un drone de modelo Eachine Wizzard220s Figure[1].

#### **3.2.1. Hardware**

Listado con el tipo de componentes que incluye en cada parte del Drone y sus características mas importantes en cuanto a autonomía, tipo de vuelo etc.

Componentes con los que esta montado:

Motores	Eachine MN2206 2300KV
Esc	Eachine 30A BLHELI_S 4 en 1 ESC
Controladora	Omnibus F4 V3 controlador de vuelo
Frame	Eachine Wizard X220S
Tx de vídeo	5,8G 72CH 25MW 200mw 600MW Transmisor conmutable
Emisora	Flysky FS-i6X 2,4GHz 10CH
Receptor	Flysky FS-A8S receptor
Cámara FPV	Eachine 1179 800TVL CCD
Antena de vídeo	Circular pagoda
Hélices	Eachine 5051 Tripala
Batería	Eachine 1500MAH 4S 14,8V 75C

Cuadro 2: Carácteristicas del Drone.

En específico, el control del drone Figure[2] se trata de un modelo del radio control: Transmisor RC Flysky i6X FS-i6X 2.4GHz 10CH AFHDS 2A con receptor X6B / IA6B / A8S para FPV RC Drone - Modo 2 (acelerador de mano izquierda) i6X + ia6B. Este control permite conectarse a una computadora mediante el cable de entrenador, y una vez conectado se usa un programa que es un simulador de vuelo virtual usando el control y de esa manera acostumbrarse a los controles.

Internamente el control contiene un microprocesador que se encarga de hacer las lecturas analógicas a digitales de ambos joysticks, sus 4 switch de 2 estados

y su switch de 3 estados, esto sin contar los botones que usa el control para la configuración:



Figura 15: Vista del control abierto

El control se alimenta con 4 baterías AA conectadas en paralelo para sumar voltaje (cada una 1.2 v), resultando en un voltaje de operación de 4.8VDC. Lo que se busca es controlar el control, para esto solo basta con suplantar la lectura analógica de los joysticks con un programa que emule estas señales cuando se requiera, siendo estos los más importantes en el control.

Otro componente importante para el control del drone es su tarjeta de vuelo:



Figura 16: Tarjeta de Vuelo.

Esta tarjeta se encarga de recibir la señal de radio del radio control y la decodifica para generar una señal de control que será interpretada por el programa de un software llamado Betaflight.

Ya que lo que se busca es que el drone opere de manera autónoma, es necesario poder ver con la cámara del drone en tiempo real.



Figura 17: Cámara del Drone

La cámara que incorpora el drone permite la transmisión de video hasta 1 km en linea recta, se encuentra cólocada en la parte frontal como se puede observar en (Figura[1]) y con ayuda de un dispositivo receptor de video por USB Figure[3], podemos usar la cámara del drone como si se tratara de una webcam.

### 3.2.2. Software

El programa con el que opera el drone es Betaflight. Es un programa que se ejecuta en modo gráfico con html. Sirve para realizar cambios a la configuración de fábrica del drone, así como el estado actual del cada uno de los sensores del drone como el acelerómetro, indicador de voltaje en la batería, estado de los motores, etc.

Betaflight es un software de controlador de vuelo (firmware) utilizado para volar embarcaciones de múltiples rotores y embarcaciones de ala fija. Esta horquilla difiere de Baseflight y Cleanflight en que se enfoca en el rendimiento de vuelo.

Para usarse, se debe habilitar una extensión en google chrome que se encuentra en la siguiente url:

<https://chrome.google.com/webstore/detail/betaflight-configurator/kdaghagfopacdngbohiknlhcocjccjao?hl=es-419>

Una vez habilitada solo se debe iniciar la aplicación y habrá iniciado.

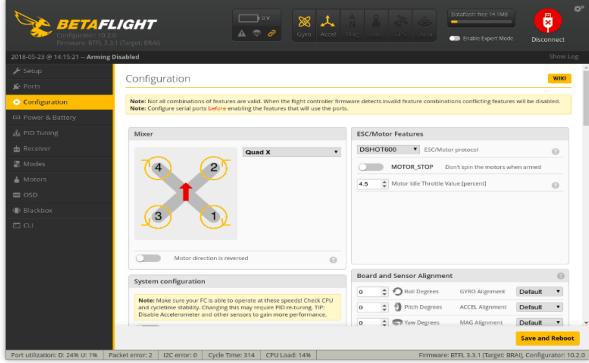


Figura 18: Software Betaflight.

### 3.2.3. Comunicación con el drone

Ya que se plantea controlar el drone, como propuesta se desarrolló un programa en python para mover los motores usando conexión USART (Universal Synchronous/Asynchronous Receiver Transmitter ó Transmisor-Receptor Síncrono/Asíncrono Universal) conectando un cable usb al drone:



Figura 19: Ubicación del puerto usb del Drone.

El drone responde a una serie de comandos para operar como una terminal llamada CLI, la cual es accesible mediante el software de Betaflight Figure[18].

Para lograr esa comunicación se planteó el siguiente procedimiento:

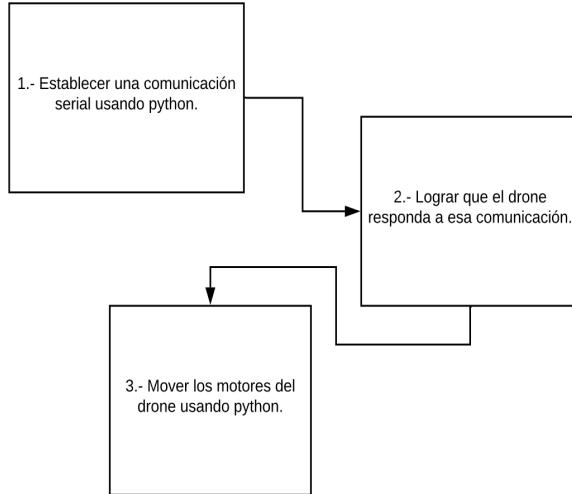


Figura 20: Planteamiento para establecer comunicación.

1.- La comunicación serial con python se puede lograr usando una biblioteca externa llamada pyserial, y se instala con pip usando:

```
$ python -m pip install pyserial
```

Con el fin de optimizar el proceso de comunicación se utilizó la terminal COM serial que proporciona el IDE de Arduino, este programa se instaló en una computadora aparte, ya que la microcomputadora JetsonNano (Figura[4]) no cuenta con una unidad de almacenamiento muy grande (64gb).

El programa se instaló de acuerdo al procedimiento descrito en la pagina: <https://www.arduino.cc/en/guide/linux> .

Como prueba se realizó lo siguiente:

- Se conectó el drone a la computadora donde se descargó arduino y se le otorgaron permisos para la comunicación serial:

```
$ ls -l /dev/ttyACM*
$ sudo chmod 777 /dev/ttyACM0
$ sudo usermod -a -G dialout USER
```

- Se seleccionó el puerto del drone “/dev/ttyACM0” dentro de arduino y se abrió el monitor serial.

No se pudo realizar una comunicación con el drone, ya que no respondía

a ningun carácter enviado.

Las características de la comunicación fueron las siguientes:

Velocidad de comunicación	115200 baud (velocidad marcada en el software Betaflight)
Puerto	/dev/ttyACM0 (puerto USB 3.0, 11.83 Mb/s)
Mensaje enviado	Carácteres aleatorios
Respuesta	Ninguna

Cuadro 3: Carácteristicas de la comunicación.

- Para solucionar el problema anterior, se escribió un programa en arduino que emula al drone en la comunicación con el programa Betaflight.  
Las características del programa se encuentran en el siguiente diagrama de flujo:

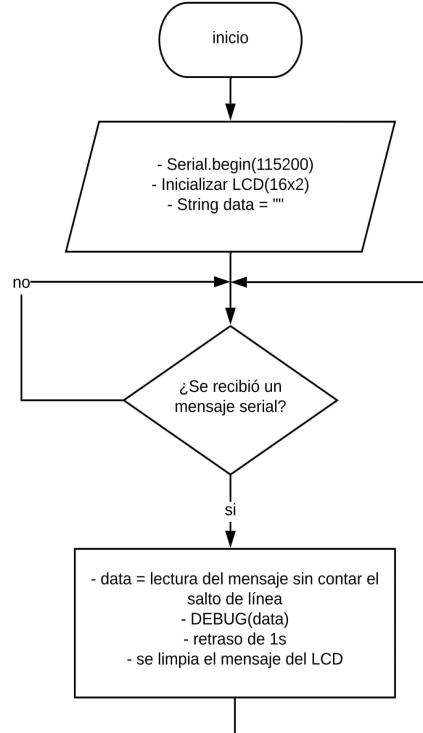


Figura 21: Proceso "main" del programa para la comunicación serial.

El flujo del programa anterior necesita de una subrutina llamada DEBUG(), la cual se encarga de mostrar en un LCD. El diagrama de flujo correspondiente se muestra a continuación:

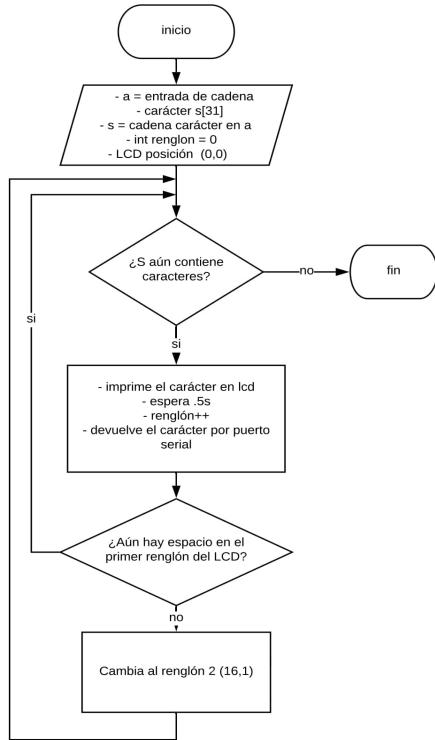


Figura 22: Diagrama de flujo subrutina DEBUG.

Ambos diagramas de flujo pertenecen al mismo programa, el cual se cargó en un arduino genuino uno y se instrumentó el hardware de la siguiente manera:

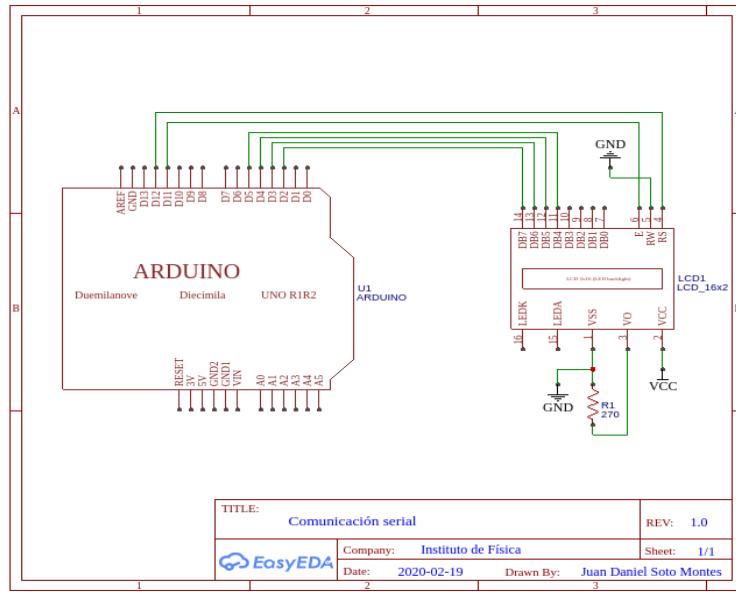


Figura 23: Esquematico de comunicación serial LCD.

Usando el monitor serial de arduino se probó el programa de la comunicación serial (Figura[21]). En esta prueba se enviaron cadenas de caracteres al microcontrolador arduino y se verificó el correcto funcionamiento del programa. Se probaron las siguientes cadenas:

Cadena	Serial — LCD
Hola	Hola — Hola
12#i-/.	12#i-/. — 12#i-/.

Cuadro 4: Carácteristicas de la comunicación.

Una vez comprobada la funcionalidad del programa, se conectó el arduino a la computadora y dentro del programa Betaflight se enlazó el puerto con el arduino (“/dev/ttyUSB0”) con una comunicación de 115200 baudios y al intentar conectar desde Betaflight, el LCD mostró el mensaje “\$M<” en 3 repeticiones separadas por 1s.

El mensaje “\$M<”, se considera como mensaje de inicialización de comunicación serial desde el monitor serial de arduino, se procedió a enviar el mensaje del programa Betaflight.

Esto se realizó seleccionando el puerto /dev/ttyACM0 que corresponde al puerto del drone y se estableció el monitor serial a una velocidad de conexión de 115200 baudios y se envió la cadena “\$M<” a lo cual el drone

no respondió con ningún mensaje, lo que significa que hasta este punto la comunicación con el drone no se ha logrado.

Al no poder entablar comunicación con el drone, se inspeccionó el programa Betaflight, se encontró una referencia a una wiki de los desarrolladores del software en:

*<https://github.com/betaflight/betaflight/wiki>*

Haciendo uso del foro de discusión, se obtuvo la información de que con el carácter “#” se podía iniciar la comunicación a la terminal CLI del drone. Se envió este carácter a través del monitor serial de la IDE de arduino al drone, y el drone respondió con el siguiente mensaje:

*EnteringCLIMode, type'exit'toreturn, or'help'*

El recibir este mensaje indica que la comunicación a sido un éxito, ya que se envió un mensaje, se recibió y se envió una respuesta y esta se recibió exitosamente.

2.- Ahora para emular esta comunicación lograda con python, se escribió un programa en python cuyo diagrama de flujo es el siguiente:

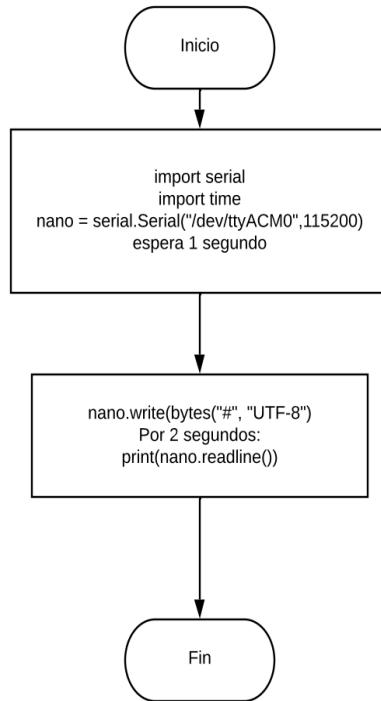


Figura 24: Diagrama de flujo de comunicación serial con python.

Este programa de python usa la biblioteca serial y time, donde serial sirve para enviar mensajes a través de los puertos COM de la microcomputadora (USB) y el módulo time sirve para realizar operaciones usando el reloj proporcionado por la computadora.

Una vez escrito el programa, se ejecutó en la terminal y como resultado se obtuvo lo siguiente:

```
(base) christian@msiPro:~/Documents/Daniel/comunicacion$ python pythonSerial.py
b'\r\n'
b"Entering CLI Mode, type 'exit' to return, or 'help'\r\n"
b'\r\n'
```

Figura 25: Comunicación lograda.

Como se puede observar, el mensaje leído por el programa indica que la comunicación usando python funcionó correctamente.

3.- El objetivo de este punto es mover los motores del drone usando python para controlar la terminal CLI del drone. Investigando cómo operar la terminal CLI

del drone, se encontró que el comando:

```
$ motor <index> [value]
```

Donde index es el número del motor, cuyas posiciones por index se muestran en la vista previa del programa Betaflight (Figura[18]) y value el valor de la velocidad del motor, la cual vale mínimo 1000 y máximo 2000.

### 3.2.4. Telemetría

Se cambió el enfoque del control del drone, para esta vez realizar el control del drone manipulando su radio-control (Figura[2]).

Al observar el circuito del control (Figura[15]), se encontró para que el control pueda funcionar necesita de 4 baterías AA conectadas en paralelo para sumar voltaje obteniendo un voltaje de operación de 4.8Vdc.

Ya que lo que se busca es controlar el control basta con suplantar la lectura analógica de los joysticks; una propuesta es usar un microcontrolador que produzca una señal analógica para que pueda emular las señales que producen los joystick y de esta manera convertir el control en uno programable.

Y para conseguir cumplir estos objetivos, se propuso lo siguiente:

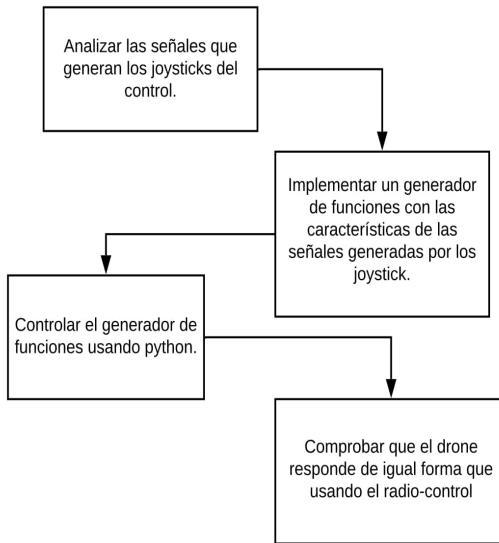


Figura 26: Diagrama de implementación de control con telemetría.

Usando arduino, se programó un analizador de funciones para identificar cuales

son las características de las señales generadas por el joystick. Para esto se generó el siguiente programa en arduino:

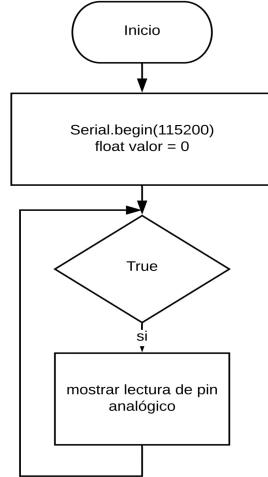


Figura 27: Diagrama de flujo Analizador de funciones.

El programa únicamente monitorea el voltaje suministrado a un pin capaz de realizar una conversión analógica-digital se obtuvieron los siguientes resultados:

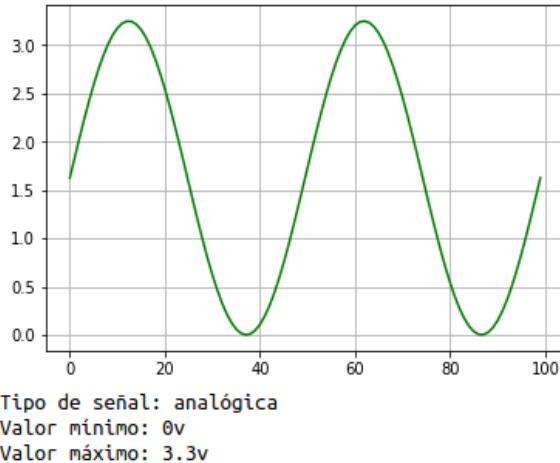


Figura 28: Carácteristicas de la señal del joystick.

Cada joystick genera dos señales con esas características, al ser 2 joysticks, obtenemos que debemos emular 4 canales. Para referencias futuras, la distribución de los canales se encuentran dados en la siguiente imagen:



Figura 29: Distribución de canales.

El resultado de analizar la señal generada por los joysticks esta en un rango de 0-655 (valores análogicos de arduino que van de 0 a 1024). Que tienen la siguiente equivalencia  $655 = 3.19V_{DC}$  y  $0 = 0V_{DC}$ , estos rangos son los mismos para cada canal del joystick y que se puede ver en la figura anterior.

Para hacer el generador de señales, se utilizó un microcontrolador Arduino Mega al cual se le cargó un programa cuyo diagrama de flujo es el siguiente:

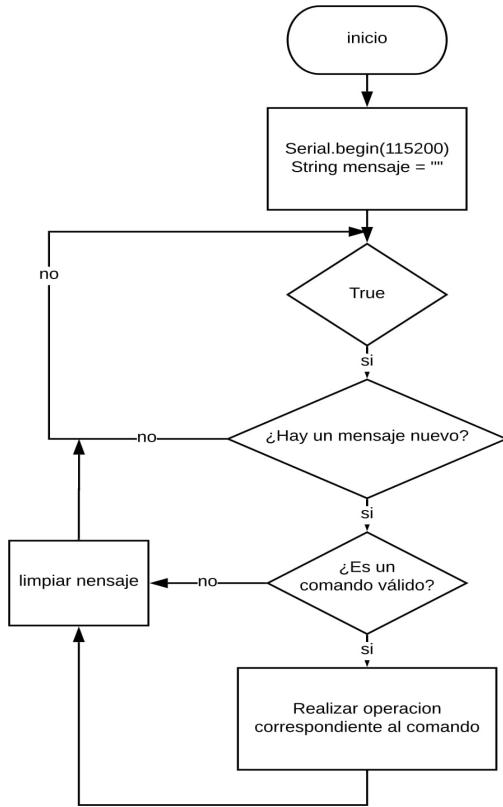


Figura 30: Diagrama de flujo generador de señales.

Se decidió usar el microcontrolador Arduino Mega por sus siguientes características:

Controlador	pines PWM	Frecuencia
Arduino Mega	2-3, 44-46	490Hz (4y13:980hz)

Cuadro 5: Características de la comunicación USART en Arduino Mega.

Como se observa en la tabla, la velocidad de comunicación que tiene este microcontrolador es suficiente para realizar los cambios en el generador de funciones para controlar al drone en tiempo real.

Al conectar esta señal al control se notó mucha interferencia y al observar la señal usando un analizador de señales, se notó que la señal tenía forma cuadrada, por lo tanto es una señal modulada por ancho de pulso (PWM).

Ya que lo que buscamos generar es una señal analógica, una señal de PWM no sirve. -Para solucionar esto, se filtró la señal usando un circuito RC:

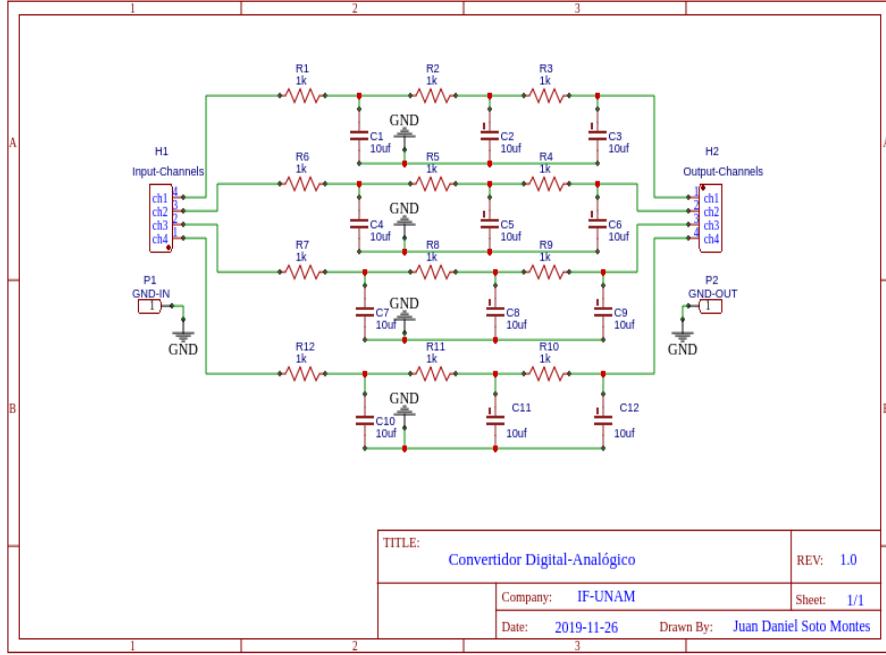


Figura 31: Circuito DAC.

Una vez que la señal ya fue filtrada, se conectó a las terminales que corresponden a los canales del drone para los ejes de cada joystick (Figura[29]).

Se probó la señal conectándolo en los canales 3 y 4 (un canal a la vez) y se comprobó que la señal sí funciona usando la pantalla en el control. Si queremos ver el valor del generador de funciones leído por el control, debemos acceder al menu "Display" del control de la siguiente manera:

- Encender el control.
- Mantener presionada la tecla OK hasta que cambie a menú.
- Elegir la opción configuración y aceptarla con OK.
- Dentro de configuración elegir la opción Display y aceptar con OK.

Este apartado del menú del control indica los valores de los canales que usa para controlar el drone, donde de los canales 1 al 4 corresponden a ambos joysticks, que es lo que nos interesa controlar con python.

Es conveniente ver esta parte del menú, ya que muestra de manera gráfica los

valores de los canales y nos sirve como referencia para saber si el generador de funciones esta generando una señal analógica y no una señal de PWM:



Figura 32: Lectura de canales del control SKYFLY.

Usando un programa en arduino con el diagrama de flujo mencionado anteriormente (Figura[30]), se configuró el generador de funciones para los 4 canales que se requieren para controlar los joysticks del control.

Con el fin de identificar los pines correspondientes a los joysticks en el circuito del control, la siguiente imagen muestra la ubicación de cada canal, que serían los jumpers que se desconectaron de los joysticks para usar la señal generada con el microcontrolador:

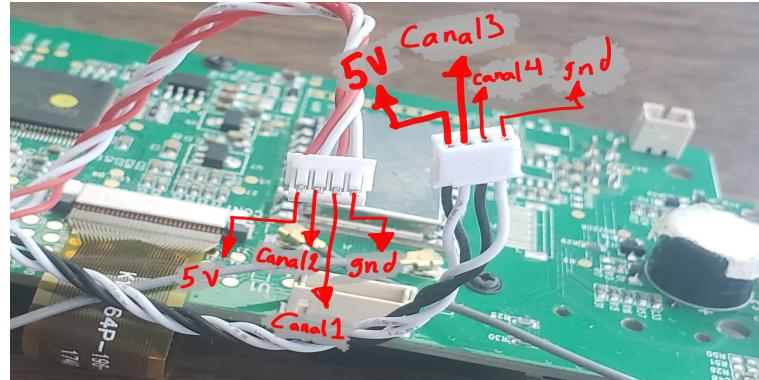


Figura 33: Canales en el circuito.

Esta parte del circuito se encuentra abriendo el control, y está situado en la parte anterior análoga de los joysticks en el circuito.

Ya que se debe hacer una combinación de movimientos con el joystick izquierdo para enlazar el control con el drone, los valores son los siguientes:

Canal	Valor
1	84
2	84
3	0
4	168

Cuadro 6: Carácteristicas de la comunicación USART en Arduino Mega.

Se debe mantener ese estado por al menos 200 ms. Si no se mueven los motores usando el control después de 2s de enlazar el control, por seguridad el drone se desconecta del control.

Los valores usados equivalen a una relación de la señal generada con PWM que van de 0 a 255 es decir  $2^8$  estados lógicos del microcontrolador que para este caso son 256 estados diferentes de señal de PWM contando el estado 0.

Ya que el control solo acepta un voltaje máximo de  $3.3V_{DC}$ , la equivalencia sería un valor de máximo de 168 de PWM para el generador de señales y  $0V_{DC}$  como valor mínimo para el mismo.

Basado en la imagen donde muestra la relación de los canales en los joysticks del control (Figura[29]), la siguiente tabla muestra el movimiento que corresponde al canal de acuerdo a sus valores:

Canal	Valor	Movimiento
1	0-83	Pitch antihorario
1	84	Ningún movimiento
1	85-168	Pitch horario
2	0-83	Roll horario
2	84	Ningún movimiento
2	85-168	Roll antihorario
3	0	Ningún movimiento
3	1-168	Elevación
4	0-83	Yaw antihorario
4	84	Ningún movimiento
4	85-168	Yawhorario

Cuadro 7: Movimientos correspondientes a cada canal y su valor.

Para ejemplificar a qué movimiento corresponde Yaw, Pitch y Roll, tenemos la siguiente imagen:

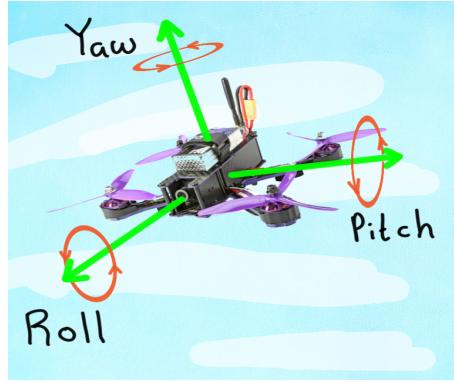


Figura 34: Movimientos básicos del dron.

Además de esto se creó una interfaz que sirve para controlar al dron mediante el puerto serial y que la computadora jetson controle al control del dron.

Ya que el programa del microcontrolador que hace la función de un generador de funciones requiere de comandos que recibe del puerto serial para generar la señal, el programa en python debe generar palabras que usan la sintáxis:

**chx-[value]**

Donde value puede tener los límites  $0 \leq value \leq 168$  y x es el canal al que se requiera cambiar el valor que va del 1 al 4. El microcontrolador también admite dos comandos extra que son **connect** y **disconnect**, al usar estos comandos se genera una secuencia de movimientos que conectan y desconectan el control del dron respectivamente.

Usando el código de python que envía mensajes al puerto serial (Figura[24]), se enviaron los comandos para controlar al dron y funcionó correctamente.

### 3.2.5. Visión artificial

Ya que el objetivo es construir un dron autónomo, un aspecto importante es el reconocimiento de su entorno. Para el caso del dron, el sensor encargado de hacer esto es la cámara, por lo tanto acceder al video de la cámara en tiempo real es muy importante para la navegación autónoma.

Gracias al receptor de video que se mostró anteriormente (Figura[3]), se puede usar la cámara del dron de manera remota como si se tratara de una webcam. Usando una biblioteca externa de python llamada opencv, se puede acceder al

video de las webcam conectadas a la computadora y usarse en el código.

Por lo que se plantearon las siguientes tareas:

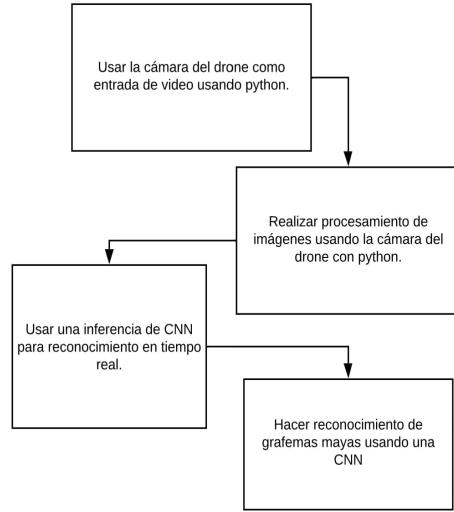


Figura 35: Diagrama de implementación de vision artificial.

La biblioteca externa opencv sirve para el tratamiento de imágenes. Para instalarlo con anaconda se usa el siguiente comando:

```
$ conda install -c conda-forge/label/cf201901 opencv
```

La cámara del drone es accesible usando python conectando la antena de video (Figura[3]) mediante puerto usb a la computadora. La función **cv2.VideoCapture()** permite a python usar una entrada de video, si el argumento es un número entero, la entrada de video será una cámara conectada a la computadora, y el valor del número entero corresponderá al índice de cámaras en la computadora.

Es decir, si se usa en una laptop, el índice 0 corresponde a la cámara de la laptop y el 1 corresponde a una cámara conectada por usb.

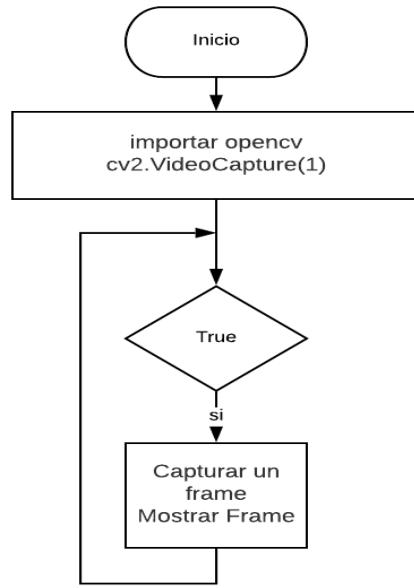


Figura 36: Diagrama de flujo que para usar la cámara del drone.

Con el fin de realizar reconocimiento de objetos, se siguieron los tutoriales de la página de opencv:

[https://docs.opencv.org/master/d9/df8/tutorial\\_root.html](https://docs.opencv.org/master/d9/df8/tutorial_root.html)

Con el material visto en los tutoriales, se aprendieron a usar algunas funciones de opencv, en específico filtros de imágenes que son útiles para el reconocimiento de objetos usando CNN.

Se implementaron códigos usando python para la detección de bordes por cambio de intensidad, movimiento por supresión de fondo, detector de bordes canny, entre otros.

Usando una inferencia de una CNN que detecta rostros con plantillas hardcascade que se obtuvieron de la página de github:

<https://github.com/opencv/opencv/tree/master/data/haarcascades>

El video de la cámara del drone se usó como entrada al programa en python y al presentar rostros frente a la cámara, se reconocieron de manera correcta.

El diagrama de flujo del programa es el siguiente:

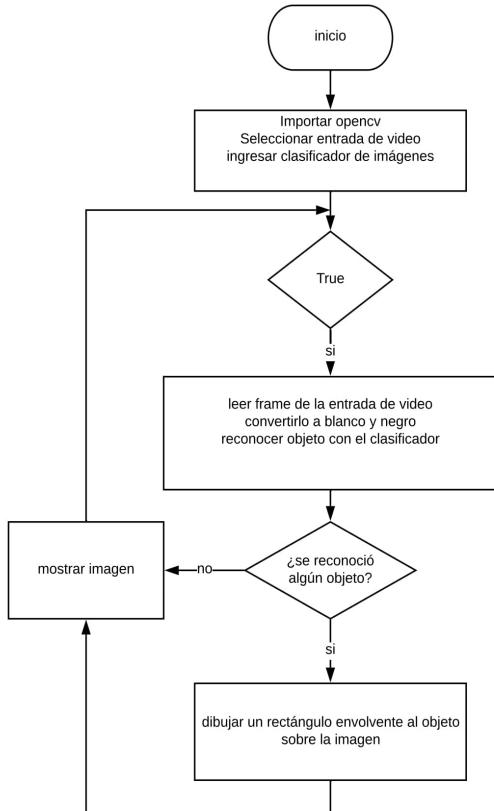


Figura 37: Diagrama de flujo para el reconocimiento de caras con opencv.

Ya que para el momento en que se realizaron estas actividades no se contaba con una base de datos completa de reconocimiento de grafemas de la escritura maya, los objetivos de esta sección (Figura[35]) se dieron por concluidos.

### 3.2.6. Vuelo autónomo

En esta sección se unen todas las actividades realizadas hasta ahora, ya que para la navegación autónoma requerimos de poder programar cada acción del drone en tiempo real, así como la manipulación de las imágenes extraídas del drone también en tiempo real.

Para lograr el objetivo de vuelo autónomo se propuso lo siguiente:

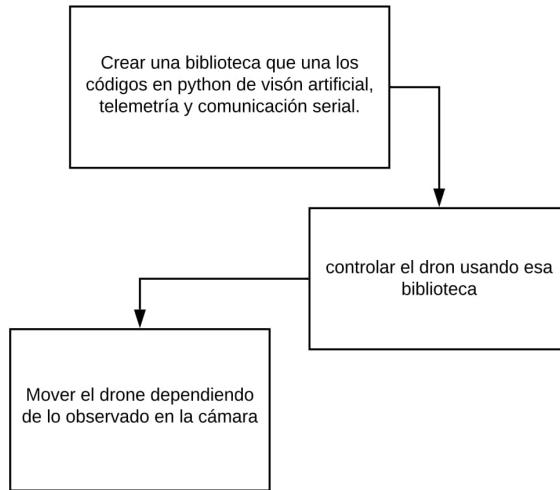


Figura 38: Diagrama de actividades a realizar para un vuelo autónomo.

El primer apartado de los procesos se completó, y la biblioteca creada se nombró "smart", esta biblioteca se subió a una cuenta de github y el código se puede ver entrando en la página:

<https://github.com/JuanDanielSoto/droneAut-nomo/blob/master/smart.py>

Esta biblioteca une cada aspecto mencionado en el primer apartado del diagrama (Figura[38]), también se creó el código main.py desde el cual se importa la biblioteca y se realiza el segundo punto.

Para el último punto de este apartado, fue necesario construir funciones extras en la biblioteca "smart" para realizar el movimiento autónomo del dron.

La inferencia proporcionada por opencv para detectar rostros funciona de la siguiente manera:

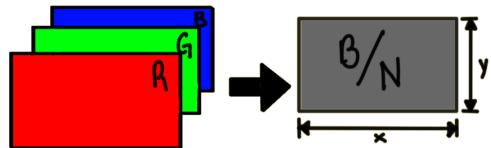


Figura 39: Tratamiento de inicial de la imagen.

Tal como lo muestra la imagen anterior (Figura[39]), la imagen de entrada es una imagen RGB, esto quiere decir que son 3 imágenes superpuestas una sobre otra y cada una es de color diferente, Rojo (R), Verde(G), Azul (B) y tiene un total de  $x$  pixeles de ancho y  $y$  pixeles de altura. Usando un filtro de imágenes, se convierte esa imagen en blanco y negro, y mantendrá el mismo ancho y alto que la imagen inicial.

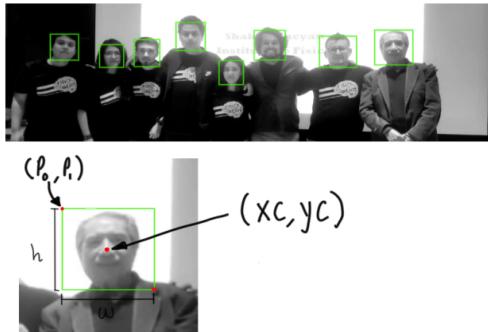


Figura 40: Encontrando el centro del objeto.

Si se quiere encontrar la coordenada del centro de una cara, tal como se muestra en la (Figura[40]), entonces se deben conocer la coordenada de la esquina superior derecha, la cual es proporcionada por la función `cascade.detectMultiScale()` de la biblioteca de opencv. Usando los valores de esa coordenada, podemos calcular la coordenada del centro de la cara ( $xc, yc$ ):

$$xc = P_0 + \frac{w}{2}$$

$$yc = P_1 + \frac{h}{2}$$

Una vez que se calculó el centro del objeto detectado (en este caso una cara), usando las herramientas de opencv se dibujó un diagrama de posiciones, el cual servirá como ayuda visual para el movimiento autónomo del drone:

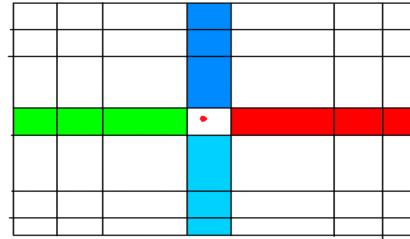


Figura 41: Diagrama de posiciones.

De acuerdo al diagrama de movimientos mostrado anteriormente, dependiendo del punto calculado como el centro del objeto, el objetivo siempre será mantener el punto en el centro de la imagen, de tal forma que el dron tendrá diferentes acciones dependiendo de esa posición:

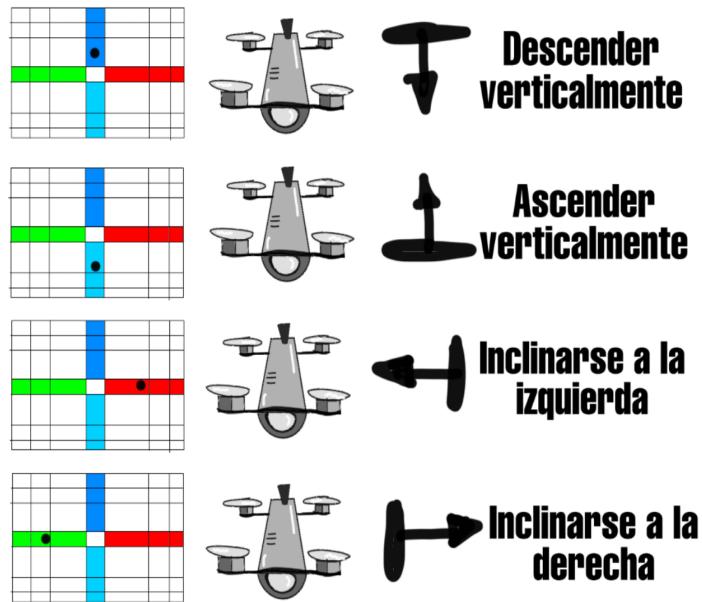


Figura 42: Diagrama de movimientos del dron.

Una vez que el dron se encuentra en el centro, no realizará ningún movimiento.

Hasta ahora sólo se tiene contemplado el movimiento para centrar un solo objeto, pues si más de un objeto es detectado, habrían muchas órdenes ejecutándose

al mismo tiempo, lo cual ocasionaría un problema para el vuelo autónomo.

Para resolver este problema, se planteó realizar una regresión lineal con los puntos de los centros de los objetos detectados, y así encontrar el centro de todos los objetos.

Solucionando así el problema, además de que todos los objetos se enfocarían de la manera más centrada posible. Viéndolo de manera gráfica, el planteamiento es el siguiente:

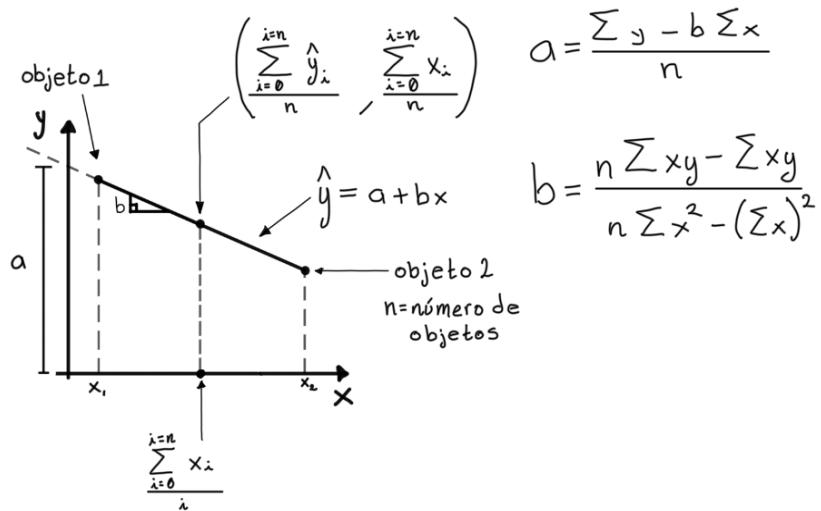


Figura 43: Cálculo del centro de una regresión lineal.

Tal como se muestra en la imagen (Figura[44]), trazar una recta que se ajuste de la mejor manera a todos los objetos detectados en una buena solución, ya que si encontramos el centro de esa recta, ahora será el punto que deberá centrarse en el diagrama de movimientos del drone (Figura[41]).

De tal forma que si usmamos la imagen con los objetos detectados (Figura[40]) y ahora aplicamos la regresión lineal, obtendremos lo siguiente:

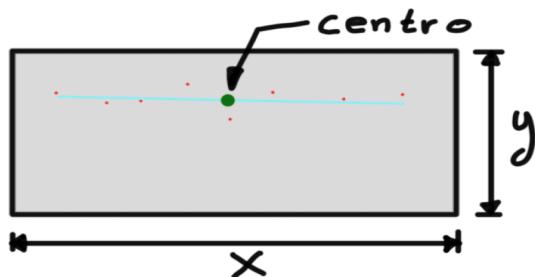


Figura 44: Centro de los objetos calculado con una regresión lineal.

Una vez que esta listo el reconocimiento, como medida de precaución para el movimiento del drone, cada vez que se le da una orden al drone, se guarda el estado de los motores en una caja negra, la cual se llama *log.txt* y se encuentra dentro de la carpeta topic que es donde se encuentra el código main del drone escrito en python.

La caja negra se compone de 4 columnas separadas por " ; " y cada columna representa 1 canal de control del drone. La primer columna corresponde al primer canal, la segunda al segundo canal y así sucesivamente.

La implementación de este archivo tiene importancia en el código, ya que si no existiera, el drone podría hacer movimientos muy bruscos, ya que siempre estarán variando los valores de los canales.

Una primer prueba se realizó usando el drone sin hélices, se ejecutó el programa y el control realizó todo respecto a lo planeado. Como observación, al subir del 60 % de la capacidad de elevación del drone, el drone vibró mucho por lo que el lente de la cámara comenzaba a retirarse, además de que el consumo de potencia de los motores inyectaba ruido en la transmisión de video.

En una prueba realizada con las hélices, el drone al despegar no fue estable, siendo la causa un error en la acción del drone al elevarse, ya que no todos los

motores giraban a la misma velocidad a pesar de que se había logrado realizar esta tarea con anterioridad, pero sin las hélices.

Ya que para este punto no se contó con más tiempo para realizar mas pruebas con el drone, como recibir información del acelerómetro del drone usando la antena del control y de esta forma estabilizar el despegue; se optó por dar por concluido el proyecto.

### 3.3. Conclusiones

Durante el transcurso del servicio social se planteó y desarrolló una solución que debido a problemas con el hardware del drone no se pudo cumplir con el objetivo del servicio social. Sin embargo se abordaron temas que abarcan la electrónica, telemetría, inteligencia artificial, reconocimiento de objetos, filtrado y procesamiento de imágenes, programación de alto y bajo nivel, uso de microcomputadoras. Individualmente hubo resultados como la comunicación de un microcontrolador mediante puerto serial con python, la creación de un generador y un lector de funciones analógico, transmisión de control serial por radio, recepción de video por radio, reconocimiento de patrones exitosos, detectores de movimiento mediante supresión de fondo, la creación de una extensa biblioteca que provee de herramientas que facilitan cada una de las tareas antes mencionadas con pocas líneas de código, por lo que la interfaz de usuario es lo suficientemente óptima.

Otro aspecto importante que no se pudo llevar a cabo fue el reconocimiento de grafemas mayas, debido a que la base de datos que recopila estos grafemas aún no lista, por lo que se mantiene pendiente para un futuro, pero inconcluso para este informe.

## Referencias

[https://docs.opencv.org/master/d9/df8/tutorial\\_root.html](https://docs.opencv.org/master/d9/df8/tutorial_root.html)  
<https://betaflight.com/>  
<https://developer.nvidia.com/embedded/jetson-nano-developer-kit>  
<https://www.coursera.org/specializations/deep-learning?>

---

Dr. J. GUADALUPE PÉREZ RAMIREZ  
Investigador Titular del Instituto de Física  
Universidad Nacional Autónoma de México