



**UNIVERSIDAD ESTATAL PENÍNSULA DE SANTA ELENA**  
**FACULTAD DE SISTEMAS Y TELECOMUNICACIONES**  
**CARRERA DE ELECTRÓNICA Y TELECOMUNICACIONES**

**TRABAJO DE TITULACIÓN**  
**PROPUESTA TECNOLÓGICA, PREVIO A LA OBTENCIÓN DEL**  
**TÍTULO DE:**  
**INGENIERO EN ELECTRÓNICA Y**  
**TELECOMUNICACIONES**

**TEMA:**

“Diseño de un Sistema de Control Autónomo para estabilizar un Robot Submarino en la  
captura de imágenes y reconocimiento de especie marina Estrella de Mar mediante  
Pixhawk y Visión Artificial”

**AUTOR:**

ANDRÉS FRANCISCO TUMBACO PILAY

**TUTOR:**

ING. SENDEY VERA MSC.

La Libertad – Ecuador

2025

## **APROBACIÓN DEL TUTOR**

En mi calidad de Tutor del trabajo de titulación denominado “Diseño de un Sistema de Control Autónomo para estabilizar un Robot Submarino en la captura de imágenes y reconocimiento de especie marina Estrella de Mar mediante Pixhawk y Visión Artificial”, elaborado por el egresado Andrés Francisco Tumbaco Pilay , de la carrera de Electrónica y Telecomunicaciones de la Universidad Estatal Península de Santa Elena, me permito declarar que luego de haber orientado, estudiado y revisado, los apruebo en todas sus partes y autorizo a los estudiantes para que inicie los trámites legales correspondientes.

La Libertad, octubre del 2024

Ing. Sendey Agustín Vera González, MG

## **AGRADECIMIENTO**

Agradezco en primer lugar a Dios por permitirme vivir y culminar con esta etapa de mi vida.

En especial agradezco a mis padres por el apoyo incondicional que me brindaron durante esta etapa de mi vida, sin su ayuda tanto económica como motivacional cumplir esta meta no hubiera sido posible.

Por último, agradezco a mi tutor Ingeniero Sendey Vera por la ayuda y disponibilidad brindada durante el desarrollo del proyecto y por incentivar me a realizar este proyecto de robot submarino con visión artificial.

*Andrés Francisco Tumbaco Pilay*

## **DEDICATORIA**

A mis padres y hermana por brindarme su apoyo en todo momento durante esta etapa de mi vida y por el esfuerzo económico que realizaron para que pueda cumplir con esta meta propuesta en mi vida.

A mis familiares que me tendieron una mano durante el transcurso de mi carrera universitaria.

A mis amigos y compañeros de carrera que supieron brindar su ayuda en momentos duros durante esta etapa universitaria.

*Andrés Francisco Tumbaco Pilay*

## **TRIBUNAL DE GRADO**

---

Ing. Washington Torres Guin, Mgt.

**DECANO DE FACULTAD**

---

Ing. José Sánchez Aquino, Mgt.

**DIRECTOR DE LA CARRERA**

---

Ing. Carlos Saldaña Enderica, Mgt.

**DOCENTE DEL ÁREA**

---

Ing. Sendey Vera González, Mgt.

**DOCENTE TUTOR**

---

Ab. María Rivera González, Mgt.

**SECRETARÍA GENERAL**

**UNIVERSIDAD ESTATAL PENÍNSULA DE SANTA ELENA**  
**FACULTAD DE SISTEMAS Y TELECOMUNICACIONES**  
**CARRERA DE ELECTRÓNICA Y TELECOMUNICACIONES**

Diseño de un sistema de control autónomo para estabilizar un robot submarino en la captura de imágenes y reconocimiento de especie marina Estrella de Mar mediante Pixhawk y visión artificial.

**Autor:** Tumbaco Pilay Andrés Francisco

**Tutor:** Ing. Vera Sendey

**RESUMEN**

El presente proyecto aborda el diseño de un sistema de navegación buscando la autonomía para un robot submarino para la captura de imágenes y reconocimiento de una especie marina como la Estrella de Mar. El sistema de navegación autónomo será implementado en el robot submarino de BlueRobotics el cual fue diseñado para ser manipulado remotamente de modo manual, el robot es actualizado con el sistema operativo BlueOS que ha sido instalado en la computadora complementaria Raspberry Pi 4 y que a su vez se aplicara visión artificial para el reconocimiento de la Estrella de Mar.

Los componentes de hardware usados para el diseño del sistema de navegación autónoma son: la cabina hermética de componentes electrónicos que está compuesta de los siguientes elementos: controlador Pixhawk PX4, Raspberry Pi 4, GPS, propulsores, y los softwares utilizados son: firmware ArduSub de ArduPilot, Mission Planner y QGroundControl. En la Raspberry Pi 4 se ejecuta el sistema operativo BlueOS que permite la comunicación entre el controlador Pixhawk. El controlador Pixhawk y el firmware ArduSub correctamente parametrizados controlan los propulsores del robot submarino. Los sensores integrados al Pixhawk como el giroscopio, acelerómetro, sensor de profundidad y el GPS incorporado proporcionan datos en tiempo real, que son utilizados para ajustar continuamente los comandos a los propulsores y así mantener la dirección y estabilidad del robot submarino durante una inmersión. En este trabajo se hace una comparativa entre el uso de los softwares Mission Planner y QGroundControl para determinar las ventajas y desventajas del comportamiento del robot submarino en una misión semiautónoma.

El reconocimiento de la especie de marina se realiza mediante el uso de técnicas de visión artificial como el entrenamiento de una red neuronal convolucional utilizando un microprocesador Raspberry Pi 4, aplicando código Python y las librerías de OpenCV. El entrenamiento de la red neuronal se aplica Google Colab que proporciona memoria GPU. El entrenamiento se desarrolla a partir del modelo preentrenado YOLOv4-Tiny, que es una versión comprimida y con mayor velocidad de detección que YOLOv4. La evaluación del reconocimiento se basó en las gráficas de precisión media promedio mAP, recall y F1.

Se logró entrenar una red neuronal convolucional personalizada la cual se evaluó con la matriz de confusión que permite visualizar de manera clara las predicciones correctas e incorrectas, donde el modelo en pruebas reales alcanzó una precisión de 93.35% y recall de 96.76%, lo que nos indica que el modelo entrenado obtuvo buenos resultados. La evaluación del sistema de navegación que realiza el robot submarino se realizó calculando el error cuadrático medio comparando la trayectoria esperada vs la trayectoria real las que permiten identificar la efectividad del sistema de navegación.

Para futuros trabajos con el robot submarino se espera que se implemente un sistema de control de estabilidad utilizando un control PID avanzado el cual sea capaz de ajustar los parámetros necesarios para que el robot responda ante cualquier tipo de condiciones en la que se encuentre, además se espera que se utilicen las herramientas que no fueron utilizadas en este proyecto como el Gripper y el Sonar Ping360, que serían de gran provecho como el Ping360 que puede trabajar en condiciones de baja visibilidad para evitar obstáculos, seguir una trayectoria en combinación con un sistema de control de PID y a la localización de objetos.

**Palabras claves:** BlueOS, Raspberry Pi4, Pixhawk, Firmware, GPU, YOLOv4-Tiny, OpenCV, Google Colab.

## ABSTRACT

The present project addresses the design of a navigation system seeking autonomy for an underwater robot for image capture and recognition of a marine species such as the starfish. The autonomous navigation system will be implemented in the BlueRobotics underwater robot which was designed to be remotely manipulated manually, the robot is updated with the BlueOS operating system that has been installed on the complementary Computer Raspberry Pi 4 and that in turn will be applied artificial vision for the recognition of the starfish.

The hardware components used for the design of the autonomous navigation system are: the hermetic cabin of electronic components that is composed of the following elements: Pixhawk PX4 controller, Raspberry Pi 4, GPS, thrusters, and the software used are: Ardupilot Ardusub firmware, Mission Planner and QGroundControl. The Raspberry Pi 4 runs the BlueOS operating system that enables communication between the Pixhawk controller. The Pixhawk controller and the correctly parameterized Ardusub firmware control the underwater robot thrusters. The Pixhawk integrated sensors such as the gyroscope, accelerometer and built-in GPS provide real-time data, which are used to continuously adjust the commands to the thrusters to maintain the direction and stability of the underwater robot during a dive. In this work, a comparison is made between the use of Mission Planner and QGroundControl software to determine the advantages and disadvantages of the underwater robot behavior in a semi-autonomous mission.

The recognition of the marine species is performed by using computer vision techniques such as the training of a convolutional neural network using a Raspberry Pi 4 microprocessor, applying Python code and OpenCV libraries. The training of the neural network is applied Google Colab which provides GPU memory the training is developed from the pre-trained model YOLOv4-Tiny, which is a compressed version and with higher detection speed than YOLOv4. Recognition evaluation was based on the average mean accuracy plots mAP, recall and F1.

A custom convolutional neural network was trained and evaluated with the confusion matrix that allows to clearly visualize the correct and incorrect predictions, where the model in real tests reached a precision of 93.35% and a recall of 96.76%, which indicates that the trained model obtained good results. The evaluation of the navigation system performed by the underwater robot was carried out by calculating the mean square error



comparing the expected trajectory vs the real trajectory, which allows to identify the effectiveness of the navigation system.

For future work with the underwater robot, it is expected to implement a stability control system using an advanced PID control which is capable of adjusting the necessary parameters so that the robot responds to any type of conditions in which it finds itself. In addition, it is expected to use tools that were not used in this project such as the Gripper and the Ping360 Sonar, which will be of great benefit, such as the Ping360 which can work in low visibility conditions to avoid obstacles, follow a path in combination with a PID control system and to locate objects.

**Keywords:** BlueOS, Raspberry Pi 4, Pixhawk, Firmware, GPU, YOLOv4-Tiny, OpenCV, Google Colab.

## DECLARACIÓN

El contenido del presente trabajo de titulación es de mi responsabilidad; el patrimonio intelectual del mismo pertenece a la Universidad Estatal Península de Santa Elena

Andrés F. Tombaco

---

Tombaco Pilay Andrés Francisco

## ÍNDICE GENERAL

|   |            |
|---|------------|
| <b>APROBACIÓN DEL TUTOR .....</b>                           | <b>I</b>   |
| <b>AGRADECIMIENTO.....</b>                                  | <b>II</b>  |
| <b>DEDICATORIA .....</b>                                    | <b>III</b> |
| <b>TRIBUNAL DE GRADO.....</b>                               | <b>IV</b>  |
| <b>RESUMEN .....</b>  | <b>V</b>   |
| <b>ABSTRACT .....</b>                                       | <b>VII</b> |
| <b>DECLARACIÓN.....</b>                                     | <b>IX</b>  |
| <b>INTRODUCCIÓN.....</b>                                    | <b>1</b>   |
| <b>CAPÍTULO I .....</b>                                     | <b>2</b>   |
| <b>1. GENERALIDADES DEL PROYECTO .....</b>                  | <b>2</b>   |
| <b>1.1. Antecedentes.....</b>                               | <b>2</b>   |
| <b>1.2. Descripción del proyecto .....</b>                  | <b>3</b>   |
| <b>1.3. Objetivos del proyecto .....</b>                    | <b>4</b>   |
| <b>1.3.1. Objetivo General .....</b>                        | <b>4</b>   |
| <b>1.3.2. Objetivos Específicos .....</b>                   | <b>5</b>   |
| <b>1.4. Justificación.....</b>                              | <b>5</b>   |
| <b>1.5. Alcance del proyecto .....</b>                      | <b>6</b>   |
| <b>1.6. Metodología .....</b>                               | <b>7</b>   |
| <b>1.6.1. Investigación Bibliográfica:.....</b>             | <b>7</b>   |
| <b>1.6.2. Investigación Exploratoria:.....</b>              | <b>7</b>   |
| <b>1.6.3. Investigación experimental: .....</b>             | <b>8</b>   |
| <b>CAPÍTULO II.....</b>                                     | <b>9</b>   |
| <b>2. MARCO REFERENCIAL .....</b>                           | <b>9</b>   |
| <b>2.1. Marco Contextual .....</b>                          | <b>9</b>   |
| <b>2.2. Marco Conceptual.....</b>                           | <b>10</b>  |
| <b>2.2.1. Medio Acuático.....</b>                           | <b>10</b>  |
| <b>2.2.1.1. Flotabilidad .....</b>                          | <b>10</b>  |
| <b>2.2.1.2. Presión del agua .....</b>                      | <b>11</b>  |
| <b>2.2.1.3. Hidrodinámica .....</b>                         | <b>11</b>  |
| <b>2.2.2. Robots Autónomos .....</b>                        | <b>12</b>  |
| <b>2.2.2.1. Robot submarino .....</b>                       | <b>12</b>  |
| <b>2.2.2.2. Clasificación de los robots submarinos.....</b> | <b>13</b>  |
| <b>2.2.2.3. Robot Submarino BlueROV .....</b>               | <b>14</b>  |

|  |           |
|--|-----------|
| 2.2.3. Hardware Requerido .....  | 15        |
| 2.2.3.1. Kit BlueROV .....   | 15        |
| 2.2.3.2. Cámara .....  | 16        |
| 2.2.3.3. Control electrónico de velocidad (ESC) .....  | 18        |
| 2.2.3.4. Propulsores.....  | 19        |
| 2.2.3.5. Modulo Power .....  | 21        |
| 2.2.3.6. Fuente de alimentación .....  | 21        |
| 2.2.3.7. Regulador de Voltaje.....   | 23        |
| 2.2.3.8. Pixhawk .....   | 24        |
| 2.2.3.9. GPS .....   | 26        |
| 2.2.3.10. Raspberry Pi .....   | 27        |
| 2.2.3.11. Fathom-X.....  | 29        |
| 2.2.3.12. Correa Fathom.....   | 30        |
| 2.2.4. Software Requerido .....  | 30        |
| 2.2.4.1. Software complementario (ArduSub Companion o BlueOS) .....                          | 30        |
| 2.2.4.2. Firmware .....  | 32        |
| 2.2.4.3. Protocolo de MAVLink .....  | 32        |
| 2.2.4.4. Mission Planner .....   | 34        |
| 2.2.4.5. QGround Control .....   | 35        |
| 2.2.4.6. Python .....  | 35        |
| 2.2.5. Visión artificial .....   | 36        |
| 2.2.5.1. Detección de Objetos .....  | 36        |
| 2.2.5.2. Red neuronal .....  | 37        |
| 2.2.5.3. Red neuronal convolucional .....  | 38        |
| 2.2.5.4. YOLO .....  | 39        |
| 2.2.5.5. OpenCV .....  | 40        |
| <b>CAPITULO III .....</b>  | <b>41</b> |
| <b>3. Diseño del algoritmo de visión artificial .....</b>                                    | <b>41</b> |
| <b>3.1. Desarrollo de un modelo personalizado para detección de objetos.....</b>             | <b>41</b> |
| 3.1.2. Crear conjunto de datos.....  | 43        |
| 3.1.3. Etiquetar conjunto de datos .....   | 45        |
| 3.1.4. Entrenamiento de la red neuronal personalizada .....                                  | 49        |
| <b>3.2. Desarrollo del código de detección de la especie marina con Python y OpenCV.....</b> | <b>56</b> |
| 3.2.1. Python .....  | 57        |

|   |     |
|---|-----|
| 3.2.2. Adaptación del modelo entrenado a la Raspberry Pi 4.....   | 62  |
| Capítulo IV.....  | 65  |
| 4. Diseño y Configuración del Sistema de Navegación Autónoma..... | 65  |
| 4.1. Mission Planner vs QGroundControl.....                       | 65  |
| 4.2. Implementación del Sistema de Control.....                   | 68  |
| 4.2.1. Arquitectura del sistema de navegación.....                | 68  |
| 4.3. Configuración y Personalización.....                         | 75  |
| 4.3.1. Personalización del software BlueOS.....                   | 75  |
| 4.3.2. Calibración de los sensores de la Pixhawk.....             | 76  |
| 4.4. Análisis de Desafíos Técnicos y Soluciones.....              | 79  |
| CAPITULO V.....   | 86  |
| 5. Pruebas y Resultados.....                                      | 86  |
| 5.1. Resultado de entrenamiento del modelo Yolov4-tiny.....       | 86  |
| 5.1.1. Pruebas del modelo entrenado.....                          | 90  |
| 5.1.2. Pruebas del modelo bajo el agua.....                       | 95  |
| 5.2. Resultado del sistema de navegación autónoma.....            | 101 |
| CONCLUSIONES.....   | 110 |
| RECOMENDACIONES.....  | 112 |
| BIBLIOGRAFÍA.....   | 114 |
| ANEXOS.....   | 118 |
| Instalación del sistema operativo.....                            | 118 |
| Conexión remota a la Raspberry Pi.....                            | 121 |
| Conexión de Robot Submarino a la Estación de Control.....         | 128 |
| Configuración de Red.....   | 128 |
| Personalización del firmware de la placa Pixhawk.....             | 131 |
| Configuración del Robot Submarino BlueROV.....                    | 134 |
| Configuración de Frame.....                                       | 134 |
| Calibración de los sensores de la Pixhawk.....                    | 136 |
| Personalización de la palanca de mando.....                       | 139 |
| Configuración de seguridad.....                                   | 141 |
| Planificación de misión autónoma.....                             | 142 |
| Mission Planner.....  | 143 |
| QGround Control.....  | 147 |
| Código de navegación autónoma.....                                | 150 |
| Encapsular Robot Submarino.....                                   | 154 |

|   |     |
|---|-----|
| Prueba de vacío del robot submarino ..... | 154 |
| Resultado de Antiplagio .....             | 156 |

## ÍNDICE DE FIGURAS

|  |    |
|--|----|
| Figura 1: Diagrama de conexiones del robot submarino .....                       | 4  |
| Figura 2: Flujo Laminar y Flujo Turbulento [11].....                             | 12 |
| Figura 3: Clasificación de los robots submarinos [1]. .....                      | 13 |
| Figura 4: Robot Submarino BlueROV [14]. .....                                    | 15 |
| Figura 5: Contenido del Kit BlueROV [15]. .....                                  | 16 |
| Figura 6: Cámara Low-Light HD [15]. .....  | 17 |
| Figura 7: Cámara Raspberry Pi 2 [17].....  | 18 |
| Figura 8: Basic ESC - Controlador electrónico de velocidad [17]. .....           | 19 |
| Figura 9: Propulsor T200 [14].....   | 20 |
| Figura 10: Modulo de Potencia [19].....  | 21 |
| Figura 11: Batería de iones de litio de BlueRobotics (14,8 V, 18 Ah) [14]. ..... | 21 |
| Figura 12: Batería Turnigy LiPo (14,8V, 10 Ah) [20]. .....                       | 22 |
| Figura 13: Fuente de alimentación de 5V 6A [18]. .....                           | 23 |
| Figura 14: Pixhawk Puertos Frontales [22]. .....                                 | 25 |
| Figura 15: Pixhawk Puertos Laterales [22].....                                   | 25 |
| Figura 16: Pixhawk Pines de Entrada y Salida [22]. .....                         | 26 |
| Figura 17: Raspberry Pi 3 modelo B [24]. .....                                   | 27 |
| Figura 18: Raspberry Pi 4 modelo B [25]. .....                                   | 28 |
| Figura 19: Placa Fathom-X que va a bordo del BlueROV [14]. .....                 | 29 |
| Figura 20: Placa Fathom-X (FXTI) [14]. .....                                     | 29 |
| Figura 21: Correa Fathom [18]. .....   | 30 |
| Figura 22: Interfaz Web de BlueOS [8].....                                       | 31 |
| Figura 23: Trama de paquete del protocolo MAVLink [28]. .....                    | 33 |
| Figura 24: Interfaz de Mission Planner .....                                     | 34 |
| Figura 25: Interfaz de QGroundControl .....                                      | 35 |
| Figura 26: Descripción general de la detección de objetos de YOLO [32].....      | 37 |
| Figura 27: Estructura de una Red Neuronal [33]. .....                            | 37 |
| Figura 28: Estructura de una Red Neuronal Convolutiva [34]. .....                | 38 |

|   |           |
|---|-----------|
| <b>Figura 29: Ejemplo de estructura de una CNN [34].</b> .....  | <b>39</b> |
| <b>Figura 30: Arquitectura YOLO [36].</b> .....   | <b>40</b> |
| <b>Figura 31: Grafica de comparación de modelos YOLOv4 en Precisión Promedio al 50% y velocidad de inferencia FPS [38].</b> ..... | <b>43</b> |
| <b>Figura 32: Descarga de conjunto de datos</b> .....   | <b>43</b> |
| <b>Figura 33: Código en Python para crear conjunto de datos</b> .....   | <b>44</b> |
| <b>Figura 34: Imager Resizer - redimensión de imágenes</b> .....  | <b>44</b> |
| <b>Figura 35: Conjunto de datos de imágenes</b> .....   | <b>45</b> |
| <b>Figura 36: Instalación de LabelImg</b> .....   | <b>46</b> |
| <b>Figura 37: Interfaz de LabelImg</b> .....  | <b>46</b> |
| <b>Figura 38: Carga del conjunto de datos a LabelImg</b> .....  | <b>47</b> |
| <b>Figura 39: Creación de la etiqueta y la clase</b> .....  | <b>47</b> |
| <b>Figura 40: etiquetado de las imágenes</b> .....  | <b>48</b> |
| <b>Figura 41: Conjunto de datos con su respectiva etiqueta</b> .....  | <b>48</b> |
| <b>Figura 42: Github repositorio yolov4tinyp4</b> .....   | <b>49</b> |
| <b>Figura 43: Clonación del repositorio yolov4tinyp4</b> .....  | <b>50</b> |
| <b>Figura 44: Archivos del repositorio clonado de yolov4tinyp4</b> .....  | <b>50</b> |
| <b>Figura 45: archivos que servirán para el entrenamiento</b> .....   | <b>50</b> |
| <b>Figura 46: obj.data - número de clases que tendrá nuestra red neuronal</b> .....   | <b>51</b> |
| <b>Figura 47: Nombre de la clase que tendrá la red neuronal personalizada</b> .....   | <b>51</b> |
| <b>Figura 48: Archivo yolov4-tiny-custom.cfg</b> .....  | <b>51</b> |
| <b>Figura 49: Archivos de entrenamiento cargados a Google Drive.</b> .....  | <b>52</b> |
| <b>Figura 50: subir archivo a Google Colab</b> .....  | <b>52</b> |
| <b>Figura 51: Configuración del entorno de Google Colab</b> .....   | <b>53</b> |
| <b>Figura 52: Clonar repositorio Darknet</b> .....  | <b>53</b> |
| <b>Figura 53: Montar Unidad de Google Drive a Google Colab</b> .....  | <b>54</b> |
| <b>Figura 54: Habilitar OpenCV y GPU</b> .....  | <b>54</b> |
| <b>Figura 55: Construir red Darknet</b> .....   | <b>54</b> |
| <b>Figura 56: Copiar archivos de Drive a directorio de Darknet en Google Colab</b> ....   | <b>55</b> |
| <b>Figura 57: Crear archivos train.txt y test.txt</b> .....   | <b>55</b> |
| <b>Figura 58: Descarga de pesos de yolov4tiny – darknet</b> .....   | <b>56</b> |
| <b>Figura 59: Entrenamiento de la red neuronal personalizada.</b> .....   | <b>56</b> |
| <b>Figura 60: Crear Nuevo proyecto en PyCharm</b> .....   | <b>58</b> |
| <b>Figura 61: Comunicación entre WinSCP y la Raspberry</b> .....  | <b>63</b> |

|  |           |
|--|-----------|
| <b>Figura 62: Trasferir archivos entre la PC y Raspberry con WinSCP.....</b>                           | <b>64</b> |
| <b>Figura 63: Arquitectura general del Robot Submarino .....</b>                                       | <b>68</b> |
| <b>Figura 64: Arquitectura de la electrónica del sistema de navegación del robot submarino.....</b>    | <b>68</b> |
| <b>Figura 65: Diagrama de conexiones del robot submarino [18]. .....</b>                               | <b>69</b> |
| <b>Figura 66: Conexión de GPS a la Pixhawk .....</b>   | <b>70</b> |
| <b>Figura 67: Diagrama de funcionamiento de los módulos de comunicación de software .....</b>          | <b>71</b> |
| <b>Figura 68: Cambio de Raspberry y Pixhawk.....</b>   | <b>72</b> |
| <b>Figura 69: Conexión de motores a la Pixhawk .....</b>   | <b>73</b> |
| <b>Figura 70: Conexión entre la Raspberry y la Pixhawk [18]. .....</b>                                 | <b>74</b> |
| <b>Figura 71: Problemas en instalación de librerías en sistema operativo BlueOS ....</b>               | <b>79</b> |
| <b>Figura 72: Notificaciones de problemas en la transmisión de video.....</b>                          | <b>81</b> |
| <b>Figura 73: Configuración para habilitar la transmisión de video en la interfaz de BlueOS .....</b>  | <b>81</b> |
| <b>Figura 74: Diagrama de flujo de lógica de navegación autónoma .....</b>                             | <b>82</b> |
| <b>Figura 75: Esquema general del sistema de control.....</b>  | <b>84</b> |
| <b>Figura 76: Gráfica Roll Angular Rate.....</b>   | <b>84</b> |
| <b>Figura 77: Gráfica de precisión del modelo entrenado.....</b>                                       | <b>87</b> |
| <b>Figura 78: Gráfica de Recall del modelo entrenado.....</b>  | <b>88</b> |
| <b>Figura 79: Gráfica de precisión media promedio mAP del modelo entrenado .....</b>                   | <b>88</b> |
| <b>Figura 80: Gráfica de F1 del modelo entrenado.....</b>  | <b>89</b> |
| <b>Figura 81: Gráfica de Loss del modelo entrenado .....</b>   | <b>89</b> |
| <b>Figura 82: Prueba del modelo entrenado con el peso o weights a 1000 iteraciones</b>                 | <b>90</b> |
| <b>Figura 83: Presencia de falso positivo del modelo entrenado a 1000 iteraciones ...</b>              | <b>91</b> |
| <b>Figura 84: Prueba del modelo entrenado con el peso o weights a 3000 iteraciones</b>                 | <b>91</b> |
| <b>Figura 85: Corrección de presencia de falso positivo a 3000 iteraciones .....</b>                   | <b>92</b> |
| <b>Figura 86: Prueba del modelo entrenado con el peso o weights a 6000 iteraciones .....</b>           | <b>92</b> |
| <b>Figura 87: Prueba del modelo entrenado a 6000 iteración sin presencia de falsos positivos .....</b> | <b>93</b> |
| <b>Figura 88: Matriz de confusión de pruebas realizadas al modelo entrenado en superficie .....</b>    | <b>93</b> |
| <b>Figura 89: Pruebas del modelo entrenado bajo el agua con el robot submarino ...</b>                 | <b>95</b> |



|   |            |
|---|------------|
| <b>Figura 90: Resultados de precisión entre 52.18% hasta 90.11% sin presencia de falsos positivos.....</b>                | <b>95</b>  |
| <b>Figura 91: Gráfica de Roll y Pitch en modo estable.....</b>  | <b>96</b>  |
| <b>Figura 92: Pruebas del modelo entrenado con el robot submarino en modo estable .....</b>                               | <b>96</b>  |
| <b>Figura 93: Desplazamiento de la estrella de mar durante pruebas presenta una precisión de 59.36%.....</b>              | <b>97</b>  |
| <b>Figura 94: Pruebas del modelo entrenado bajo el agua en la noche con el robot submarino.....</b>                       | <b>97</b>  |
| <b>Figura 95: Precisión de 97.43% del modelo entrenado durante las pruebas realizadas en la noche.....</b>                | <b>98</b>  |
| <b>Figura 96: Matriz de confusión de pruebas realizadas al modelo entrenado bajo el agua.....</b>                         | <b>98</b>  |
| <b>Figura 97: Predicciones erróneas realizadas bajo el agua por el modelo entrenado .....</b>                             | <b>100</b> |
| <b>Figura 98: presencia de falsos negativos en pruebas realizadas bajo el agua.....</b>                                   | <b>100</b> |
| <b>Figura 99: Maniobras del robot submarino con el joystick en modo manual.....</b>                                       | <b>101</b> |
| <b>Figura 100: Prueba 1 de misión autónoma realizada en Mission Planner.....</b>  | <b>102</b> |
| <b>Figura 101: Trayectoria trazada de la misión autónoma realizada en Mission Planner.....</b>                            | <b>102</b> |
| <b>Figura 102: Gráfica de Roll y Pitch durante la misión autónoma realizada en Mission Planner .....</b>                  | <b>103</b> |
| <b>Figura 103: Prueba 2 de misión autónoma realizada en QGroundControl.....</b>   | <b>103</b> |
| <b>Figura 104: Trayectoria trazada de la misión autónoma realizada en QGroundControl .....</b>                            | <b>103</b> |
| <b>Figura 105: Gráfica del comportamiento de Pitch durante la misión autónoma.</b>  | <b>104</b> |
| <b>Figura 106: Prueba 3 de misión autónoma realizada en QGroundControl.....</b>   | <b>104</b> |
| <b>Figura 107: Trayectoria trazada de la prueba 3 de misión autónoma realizada en QGroundControl .....</b>                | <b>105</b> |
| <b>Figura 108: Comprobación de cumplimiento de trayectoria en el Waypoint 2....</b>                                       | <b>105</b> |
| <b>Figura 109: Gráfica de comportamiento de Roll y Pitch durante la misión autónoma realizada en QGroundControl .....</b> | <b>106</b> |
| <b>Figura 110: Gráfica tridimensional entre la trayectoria esperada vs la trayectoria real .....</b>                      | <b>107</b> |

|  |            |
|--|------------|
| <b>Figura 111: Grafica del robot submarino durante la trayectoria.....</b>                         | <b>107</b> |
| <b>Figura 112: Gráfica del Error Cuadrático medio en coordenadas X, Y y Z .....</b>                | <b>109</b> |
| <b>Figura 113: Interfaz de Raspberry Pi Imager .....</b>   | <b>118</b> |
| <b>Figura 114: Sistemas Operativos .....</b>   | <b>119</b> |
| <b>Figura 115: Otras imágenes basadas en el sistema operativo Raspberry Pi.....</b>                | <b>119</b> |
| <b>Figura 116: Interfaz Raspberry Pi Imager - Seleccionar tarjeta SD.....</b>                      | <b>119</b> |
| <b>Figura 117: Interfaz Raspberry Pi Imager - Seleccionar opciones avanzadas.....</b>              | <b>120</b> |
| <b>Figura 118: Opciones avanzadas - habilitar SSH.....</b>   | <b>120</b> |
| <b>Figura 119: Opciones avanzadas - habilitar red Wi-Fi.....</b>                                   | <b>121</b> |
| <b>Figura 120: Interfaz de Advanced IP Scanner .....</b>   | <b>122</b> |
| <b>Figura 121: Configuración de PuTTY.....</b>   | <b>122</b> |
| <b>Figura 122: Autenticación para acceso a la Raspberry Pi.....</b>                                | <b>123</b> |
| <b>Figura 123: Actualizar el sistema de la Raspberry Pi.....</b>                                   | <b>123</b> |
| <b>Figura 124: Herramienta de configuración - Interface Options .....</b>                          | <b>124</b> |
| <b>Figura 125: Herramienta de configuración – VNC.....</b>   | <b>124</b> |
| <b>Figura 126: Habilitar VNC.....</b>  | <b>125</b> |
| <b>Figura 127: Herramienta de configuración - Display Options.....</b>                             | <b>125</b> |
| <b>Figura 128: Herramienta de configuración - VNC Resolution .....</b>                             | <b>126</b> |
| <b>Figura 129: Configuración de IP para VNC Server.....</b>  | <b>126</b> |
| <b>Figura 130: Autenticación en VNC Server .....</b>   | <b>127</b> |
| <b>Figura 131: Interfaz gráfica de acceso remoto a la Raspberry Pi.....</b>                        | <b>127</b> |
| <b>Figura 132: Configuración de IP Estática a la Raspberry Pi 4.....</b>                           | <b>128</b> |
| <b>Figura 133: Configuración de IP Estática - Redes e Internet.....</b>                            | <b>129</b> |
| <b>Figura 134: Configuración de IP Estática - Centro de redes y recursos compartidos .....</b>     | <b>129</b> |
| <b>Figura 135: Configuración de IP Estática - Cambiar configuración del adaptador .....</b>        | <b>130</b> |
| <b>Figura 136: Configuración de IP Estática - USB Ethernet Realtek USB FE.....</b>                 | <b>130</b> |
| <b>Figura 137: Configuración de IP Estática - Protocolo de internet versión 4 (TCP/IPv4) .....</b> | <b>130</b> |
| <b>Figura 138: Asignación de IP Estática.....</b>  | <b>131</b> |
| <b>Figura 139: Pixhawk conectada a PC por medio de cable USB-Micro USB .....</b>                   | <b>132</b> |
| <b>Figura 140: Seleccionamos el Puerto COM6 ArduPilot (COM6).....</b>                              | <b>132</b> |
| <b>Figura 141: Instalación de Firmware Sub en Mission Planner.....</b>                             | <b>132</b> |

|  |            |
|--|------------|
| <b>Figura 142: Seleccionar plataforma a utilizar Pixhawk 1 .....</b>                                     | <b>133</b> |
| <b>Figura 143: indicaciones a seguir para instalar el firmware Sub en la Pixhawk .</b>                   | <b>133</b> |
| <b>Figura 144: Interfaz de Mission Planner con firmware instalado .....</b>                              | <b>134</b> |
| <b>Figura 145: Seleccionamos Frame BlueROV2/Vectored .....</b>   | <b>135</b> |
| <b>Figura 146: Configuración de giro de los motores .....</b>  | <b>136</b> |
| <b>Figura 147: Cambio de orientación en Roll90 para calibrar la Pixhawk.....</b>                         | <b>136</b> |
| <b>Figura 148: Calibración de acelerómetro de la Pixhawk.....</b>  | <b>137</b> |
| <b>Figura 149: Calibración de acelerómetro – movimiento del robot en las direcciones indicadas .....</b> | <b>137</b> |
| <b>Figura 150: Calibración de compass interno y externo GPS .....</b>                                    | <b>138</b> |
| <b>Figura 151: Calibración de joystick.....</b>  | <b>139</b> |
| <b>Figura 152: Configuración de botones del joystick .....</b>   | <b>140</b> |
| <b>Figura 153: Configuración predeterminada de botones del joystick [43].....</b>                        | <b>140</b> |
| <b>Figura 154: Configuración de seguridad del robot submarino.....</b>                                   | <b>141</b> |
| <b>Figura 155: Interfaz de Mission Planner mostrando el robot submarino habilitado por GPS .....</b>     | <b>142</b> |
| <b>Figura 156: Interfaz en Mission Planner para Planificar Misiones Autónomas..</b>                      | <b>143</b> |
| <b>Figura 157: Crear misiones de manera automática - Dibujar Polígono .....</b>                          | <b>144</b> |
| <b>Figura 158: Auto WP – SimpleGrid .....</b>  | <b>144</b> |
| <b>Figura 159: Configuración de Auto WP – SimpleGrid.....</b>  | <b>145</b> |
| <b>Figura 160: Planificación de misiones autónomas.....</b>  | <b>145</b> |
| <b>Figura 161: Trayectoria a seguir durante pruebas .....</b>  | <b>146</b> |
| <b>Figura 162: Menú de opciones para ejecutar la misión autónoma programada..</b>                        | <b>146</b> |
| <b>Figura 163: Crear Misiones autónomas en QGroundControl .....</b>                                      | <b>147</b> |
| <b>Figura 164: Interfaz de planificación de misión en QGround Control .....</b>                          | <b>148</b> |
| <b>Figura 165: Misiones autónomas a seguir durante las pruebas en el software QGroundControl .....</b>   | <b>148</b> |
| <b>Figura 166: profundidad de despeje de la misión autónoma .....</b>                                    | <b>149</b> |
| <b>Figura 167: Configuración de parámetro Pitch .....</b>  | <b>149</b> |
| <b>Figura 168: Configuración de parámetro Roll .....</b>   | <b>149</b> |
| <b>Figura 169: Configuración de parámetro Yaw.....</b>   | <b>150</b> |
| <b>Figura 170: Colocación de grasa de silicona a las juntas tóricas .....</b>                            | <b>154</b> |
| <b>Figura 171: Bomba de vacío manual con manómetro .....</b>   | <b>155</b> |

## ÍNDICE DE TABLAS

|  |           |
|--|-----------|
| <b>Tabla 1. Especificaciones de cámara USB Low-Light HD .....</b>                                    | <b>17</b> |
| <b>Tabla 2. Especificaciones técnicas de propulsores T200 .....</b>                                  | <b>20</b> |
| <b>Tabla 3. Especificaciones batería de iones de litio de BlueRobotics (14,8 V, 18 Ah)<br/>.....</b> | <b>22</b> |
| <b>Tabla 4. Especificaciones de batería Turnigy LiPo (14,8 V, 10 Ah) .....</b>                       | <b>23</b> |
| <b>Tabla 5. Características principales del Pixhawk .....</b>  | <b>24</b> |
| <b>Tabla 6: Librerías a instalar para ejecutar la detección de Estrella de Mar.....</b>              | <b>58</b> |
| <b>Tabla 7: Características del Software Mission Planner y QGroundControl.....</b>                   | <b>66</b> |
| <b>Tabla 8: Ventajas y Desventajas del Software Mission Planner y QGroundControl<br/>.....</b>       | <b>67</b> |
| <b>Tabla 9 Conexión de GPS a la placa Pixhawk .....</b>  | <b>70</b> |
| <b>Tabla 10. Conexión de motores a la placa Pixhawk .....</b>  | <b>73</b> |
| <b>Tabla 11. Resultados del entrenamiento de red neuronal convolucional<br/>personalizada. ....</b>  | <b>87</b> |

## LISTADO DE ABREVIATURAS

|       |   |
|-------|---|
| AUV:  | Autonomus Underwater Vehicle, Vehículo submarino autónomo                                   |
| ROV:  | Remotely Operated Vehicle, Vehículo operado remotamente                                     |
| IAUV: | Autonomus Underwater Vehicle Interventions, Vehículo submarino autónomo para intervenciones |
| IMU:  | Inertial Measurment Unit, Unidad de medida inercial   |
| CNN:  | Convolutional Neural network, Red neuronal convolucional                                    |
| GPU:  | Graphics Processing Unit, Unidad de procesamiento grafico                                   |
| CPU:  | Central Processing Unit, Unidad central de procesamiento                                    |
| YOLO: | You Only Look Once, Solo miras una vez  |

## INTRODUCCIÓN

El avance exponencial que han experimentado los vehículos no tripulados ha despertado gran interés a la Universidad Estatal Península de Santa Elena, enfocándose en el desarrollo de robot no tripulados que sean capaces de realizar misiones autónomas programadas. Debido a esto, se decidió diseñar un sistema de navegación autónoma para un robot submarino que sea capaz de reconocer una especie marina Estrella de Mar mediante visión artificial.

El proyecto se encuentra conformado por cinco capítulos; En el primer capítulo se establecen los objetivos propuestos en el proyecto, así mismo se presenta como se encuentra constituido el proyecto, el alcance que tendrá, su justificación y que metodología se emplea para el cumplimiento de los objetivos. El segundo capítulo se encuentra enfocado en los conceptos más importantes de navegación autónoma y visión artificial. También se menciona los elementos electrónicos que se utilizaron en el proyecto para el diseño de sistema de navegación autónoma y el reconocimiento de la especie marina. El tercer capítulo se centra en el diseño del algoritmo de visión artificial, donde se da a conocer los pasos a seguir para entrenar una red neuronal convolucional personalizada, además se desarrolla el código de programación para la detección de la especie marina Estrella de Mar. El capítulo cuatro contiene el diseño y configuración del sistema de navegación autónoma, en el cual se menciona la importancia del uso de los softwares Mission Planner y QGroundControl para la configuración de parámetros del robot submarino. Finalmente, en el capítulo cinco se presentan los resultados obtenidos durante las pruebas realizadas al sistema de navegación autónoma y del sistema de detección de la especie marina.

# CAPÍTULO I

## 1. GENERALIDADES DEL PROYECTO

### 1.1. Antecedentes

Los robots submarinos tienen sus inicios a principios de la década de los 50, con la construcción del robot submarino POODLE desarrollado por el francés Dimitri Rebikoff el cual era operado remotamente [1].

Un vehículo submarino autónomo, o AUV, es un vehículo submarino autopropulsado, no tripulado y sin ataduras, capaz de realizar actividades sencillas con poca o ninguna supervisión humana. Los AUV se utilizan a menudo como plataformas de estudio para cartografiar el fondo marino o caracterizar las propiedades físicas, químicas o biológicas del agua o para el reconocimiento las especies marinas [2].

Con el avance tecnológico los robots submarinos han comenzado a revolucionar la exploración del fondo marino, incentivando a universidades e institutos científicos a investigar y desarrollar sus propios prototipos de robot submarino. Como ejemplo la Universidad Estatal Península de Santa Elena, en el 2013 Eduardo Lucín desarrolló un prototipo sumergible híbrido para adquirir datos de temperatura e imágenes, sirviendo de herramienta de medición para los estudiantes de la facultad de ciencias del mar [3]; En el 2019 Carlos Montoya desarrollo un prototipo de robot submarino autónomo tipo torpedo capaz de recopilar datos oceanográficos [4]; En el 2020 Arturo Cadena y Sendey Vera desarrollaron un vehículo submarino autónomo híbrido y portátil para la exploración antártica, el cual cuenta con algoritmos de sistema de navegación inercial y Visión por computadora [5].

En el año 2008 Lorenz Meier junto con un equipo de estudiantes y con la ayuda de su profesor Marc Pollefeys iniciaron con la construcción de un hardware y software que facilite a un dron a esquivar obstáculos en espacios interiores, dando inicio de esta manera a la placa controladora de vuelo Pixhawk [6]. Al obtener buenos resultados decidió continuar con el proyecto el cual dejaron el software y hardware de código abierto proporcionando el código fuente del software de estación terrestre QGroundControl; el software PX4 con el cual se controlaba el vuelo del dron; el diseño del hardware del controlador Pixhawk; y el protocolo de comunicación MAVLink [7].

A partir del proyecto de Lorenz Meier se empezó a implementar la Pixhawk para controlar el vuelo de drones hasta en la actualidad, pero no fue hasta el 2015 que el proyecto Companion de BlueRobotics empezó a implementar el controlador Pixhawk en robots submarinos con el objetivo de transmitir video y entablar comunicación entre el robot submarino con la computadora de superficie. Actualmente BlueRobotics cuenta con varios vehículos acuáticos que funcionan con el software BlueOS, que es capaz de adaptarse a cualquier tipo de robots acuáticos, el cual reemplazo al software ArduSub Companion [8].

Para el reconocimiento de especies marinas los robots submarinos utilizan visión artificial para poder diferenciar entre distintas especies. Durante casi 60 años, científicos e ingenieros han tratado de idear formas para que las máquinas vean y comprendan datos visuales.

La visión artificial es una rama de la inteligencia artificial que permite que las computadoras y los sistemas extraigan información significativa de imágenes digitales, videos y otras entradas visuales y tomen medidas o hagan recomendaciones basadas en esa información, la visión artificial es la que permite ver, observar y comprender lo que está visualizando [9].

La tarea principal del sistema de visión artificial en un AUV es la de procesar y extraer información de las imágenes captadas para su posterior uso en la navegación del submarino. Para el reconocimiento de especies marinas reconocimiento típicamente implica la identificación de objetos de interés preestablecido en el algoritmo de procesamiento visual [10].

## **1.2. Descripción del proyecto**

El presente proyecto consiste en desarrollar un sistema de navegación autónoma para un robot submarino que permita mantenerse estable bajo el agua, utilizando el controlador Pixhawk. En el controlador Pixhawk se cargará el firmware ArduSub el cual contiene la lógica necesaria para controlar los motores, servos y luces del robot submarino.

El control de los propulsores del robot submarino se realiza a través de la combinación de hardware y software que trabajan juntos para proporcionar un control preciso. El controlador Pixhawk hace la función de cerebro del sistema, es el encargado de recibir instrucciones de movimientos desde la estación de control a través de la Raspberry Pi que actúa como intermediario entre la estación de control en la superficie y el controlador



Pixhawk, además se encarga de recibir comandos por medio del cable Tether y los envía a la Pixhawk. Los controladores electrónicos de velocidad ESCs reciben señales PWM desde el Pixhawk las cuales determinan la dirección y velocidad de los propulsores. El software de control desde la superficie es el que permite operar, planificar misiones y enviar comandos al robot submarino. El GPS y los sensores integrados al Pixhawk como el giroscopio, acelerómetro y sensor de profundidad proporcionan datos en tiempo real al Pixhawk, que son utilizados para ajustar continuamente las instrucciones a los propulsores para mantener la estabilidad y dirección del robot submarino.

Para el reconocimiento de la especie marina se entrena una red neuronal convolucional personalidad a partir del modelo preentrenado YOLOv4-Tiny, el entrenamiento de la red se realiza en Google Colab que proporciona GPU que permite entrenar la red mucho más rápido. Luego de entrenar la red neuronal convolucional se desarrolla el algoritmo que hará el proceso de detección de la Estrella de Mar el cual se integra a la Raspberry Pi del robot submarino para realizar pruebas de funcionamiento.

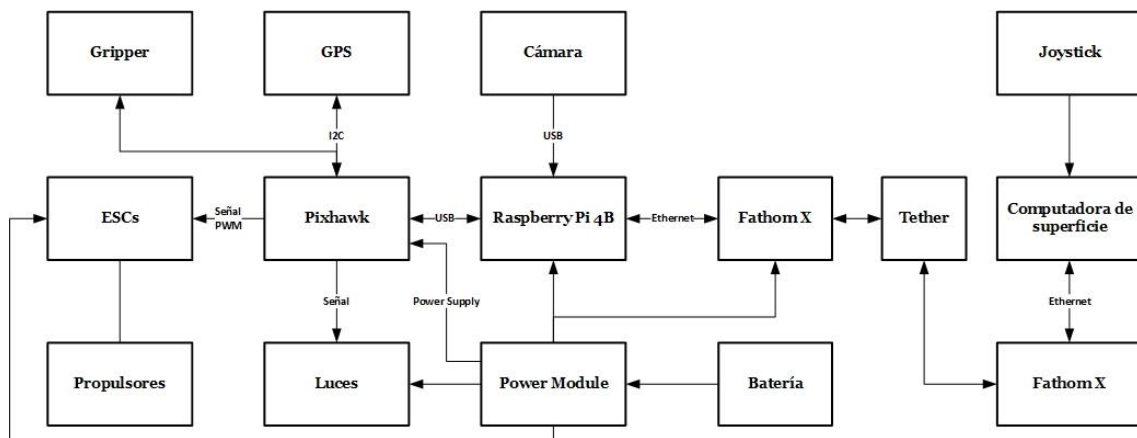


Figura 1: Diagrama de conexiones del robot submarino

Imagen elaborada por el autor

### 1.3. Objetivos del proyecto

#### 1.3.1. Objetivo General

Diseñar un sistema de control autónomo para estabilizar un robot submarino en las capturas de imágenes y reconocimiento de especie marina estrella de mar mediante uso de tecnología con Pixhawk y visión artificial.

### **1.3.2. Objetivos Específicos**

- Estudio del sistema electrónico del robot submarino para controlar sus movimientos y realización de pruebas de funcionamiento en software de simulación Mission Planner
- Establecer los parámetros necesarios en el software Mission Planner para la supervisión y estabilización del robot submarino.
- Diseñar la lógica de programación y algoritmo de control para el movimiento autónomo del robot submarino utilizando la tarjeta Pixhawk.
- Desarrollar el algoritmo de procesamiento visual para que el robot sea capaz de identificar especies marinas mediante una cámara de video incorporada al robot utilizando Raspberry Pi y OpenCV.

### **1.4. Justificación**

Los robots submarinos pueden ser herramienta importante para la investigación marina, permitiendo la exploración de hábitats en áreas que son inaccesibles para los humanos, además que cuentan con cámaras y sensores capaces de recopilar información sobre la vida marina. También pueden ayudar a crear representaciones tridimensionales de reservas marinas y ayudar a la conservación oceánica identificando especies en peligro de extinción y a identificar y recolectar desechos marinos.

Actualmente se conoce que, en la provincia de Santa Elena, se ha diseñado muy pocos robot submarinos con control autónomo y que estos cuenten con visión artificial para la identificación de especies marinas, ya sea por desconocimiento o por la complejidad de la misma, quedándose atrás con otras provincias del Ecuador y estando lejos a diferencia de otros países más desarrollados donde este tipo de tecnología es muy utilizada, obteniendo enormes beneficios en la utilización de robots submarinos.

Este proyecto se desarrolló con la finalidad de ayudar con el desarrollo de la navegación autónoma de los vehículos sumergibles y dejar una base de conocimiento a los estudiantes de la Universidad Estatal Península de Santa Elena, de la carrera de electrónica y telecomunicaciones acerca de la navegación autónoma del robot submarino utilizando el controlador Pixhawk como tecnología diferente e innovadora con respecto a otros trabajos realizados anteriormente. Así mismo se contribuirá al desarrollo de sistemas de visión artificial, proporcionando la documentación de como entrenar una red neuronal convolucional personalizada a partir de un modelo preentrenado. Para de esta manera

contribuir con el desarrollo tecnológico de la provincia y que esta tecnología se pueda utilizar para fines benéficos de la misma.

### **1.5. Alcance del proyecto**

El presente proyecto tiene como finalidad parametrizar un controlador Pixhawk para trazar la trayectoria de un robot submarino con ayuda del software Mission Planner esto ayudara a estabilizar el vehículo y poder plantear misiones autónomas.

El estudio detallado del sistema electrónico del robot submarino permitirá evaluar la funcionalidad y compatibilidad de cada componente. El controlador Pixhawk y el firmware ArduSub serán los encargados de controlar los movimientos de los propulsores del robot submarino por medio de instrucciones enviadas desde la estación de control en la superficie a través del cable tether que se conecta a la Raspberry Pi por medio de la tarjeta Fathom-X, los mensajes se envían utilizando el protocolo MAVLink. En Mission Planner se hará la configuración de parámetros de calibración de los sensores que integra el Pixhawk y del GPS que se incorpora con el objetivo de establecer ubicación del robot submarino, la calibración incluye la alineación del giroscopio, acelerómetro y magnetómetro estos parámetros son esenciales para establecer la orientación, posición, estabilizar el robot y planificar misiones autónomas por medio de puntos de referencias (Waypoint).

Para el desarrollo del algoritmo de procesamiento de imagen se utiliza técnicas de visión artificial utilizando la tarjeta Raspberry Pi y OpenCV, se entrenara una red neuronal convolucional personalizada utilizando el modelo preentrenado YOLOv4-Tiny, la red entrenada realizara el proceso de detección de la especie marina Estrella de Mar en tiempo real, además se realizaran pruebas del funcionamiento integrando el algoritmo de detección con la cámara de video y las Raspberry Pi, primero se realizara pruebas fuera del agua para comprobar la precisión de la red neuronal antes de implementar el sistema de visión artificial al robot submarino.

Las pruebas de funcionamiento del sistema de navegación y de visión artificial del robot submarino se realizarán en piscina y no se harán pruebas en el mar por motivos de costos. Tampoco se realizará un análisis estadístico de los resultados de navegación, las pruebas en el sistema de navegación se centraran en cumplimiento de la misión autónoma planificada realizada por el robot submarino.

## **1.6. Metodología**

La metodología es el campo que estudia el conjunto de técnicas o métodos que se utilizan en la investigación para lograr un objetivo establecido. La metodología de un proyecto contiene todos los pasos y procedimientos que se deben realizar para llevar a cabo un proyecto determinado.

Los pasos a seguir para llevar a cabo la propuesta presentada para diseñar un sistema de navegación autónoma para el robot submarino utilizando el controlador Pixhawk y que a su vez sea capaz de identificar una especie marina “Estrella de Mar” por medio de visión artificial, primero se da comienzo por la etapa de investigación bibliográfica, exploratoria y experimental.

### **1.6.1. Investigación Bibliográfica:**

Este tipo de investigación es necesaria con el fin de obtener la mayor cantidad de información que se va a utilizar, la encontramos en tesis, informes, proyectos, artículos, revistas científicas, etc. La documentación indagada se encuentra relacionada con proyectos usando el controlador Pixhawk donde es necesario saber cómo funciona el controlador Pixhawk que es el elemento principal robot submarino, de qué manera se encuentra constituido el robot de BlueRobotics tipo BlueROV que es el robot con el que trabajaremos, como se realiza la comunicación entre el robot y el ordenador en la superficie y que software de simulación utilizaremos. Además, esta investigación servirá de gran ayuda en el tema de la Visión artificial del proyecto, al ser en la actualidad este tipo de tecnología temas de gran estudio veremos la forma de como entrenar una red neuronal personalizada utilizando los modelos preentrenado de YOLO.

### **1.6.2. Investigación Exploratoria:**

La investigación exploratoria es requerida en este proyecto debido a que el controlador Pixhawk que usa el robot submarino son más aplicados a vehículos Aéreos, por lo que se tendrá que indagar información de drones para aplicarla en un robot submarino e ir avanzando en el proyecto con la poca información que existe sobre cómo aplicar este tipo de controlador en robot Submarinos. También se investigarán técnicas de visión artificial para integrar este sistema al robot submarino BlueROV, esperando obtener la mayor cantidad de datos para entrenar una red neuronal convolucional y poder crear una red personalizada utilizando la Raspberry Pi y OpenCV.

### **1.6.3. Investigación experimental:**

Esta investigación busca aplicar procesos experimentales específicos sobre el sistema de navegación del robot BlueROV, tendremos que realizar modificaciones para que realice misiones autónomas y a la vez se ejecute la visión artificial, para esto es necesarios realizar pruebas experimentales tanto del sistema de navegación como el de visión artificial.

## CAPÍTULO II

### 2. MARCO REFERENCIAL

#### 2.1. Marco Contextual

Un robot autónomo es aquel que tiene la capacidad de ejecutar tareas y tomar decisiones basado en lo que percibe o para lo que fue programado ya sea para moverse dentro del entorno en que se encuentre, reconocer algún tipo de objeto, recoger objetos específicos, sin la intervención directamente del ser humano.

Los robots submarinos también conocidos como ROV por sus siglas en ingles Remotely Operated Vehicle (Vehículos Operados Remotamente) o AUV Autonomus Underwater Vehicle (Vehículos Submarinos Autónomos), están diseñados para explorar y operar en entornos acuáticos hostiles en las profundidades del océano, este tipo de robot hoy en día representan una de las innovaciones tecnológicas más interesantes con respecto a la exploración del fondo marino abriendo paso a nuevas fronteras con respecto a la investigación de los océanos y transformando la forma en que el hombre interactúa con el mundo acuático sin perjudicar el medio ambiente marino.

La Universidad Estatal Península de Santa Elena es una institución que ha incentivado a sus alumnos al desarrollo e innovación de nuevas tecnologías que contribuyan al avance de la provincia, desarrollando así varios prototipos de robot submarino. En este proyecto se va a trabajar con el robot submarino de Blue Robotics una de las principales compañías que busca habilitar el futuro de la robótica marina.

Este proyecto se realiza en la provincia de Santa Elena para seguir con el avance tecnológico de los robots submarinos siguiendo la línea de la compañía Blue Robotics la cual usa la placa Pixhawk como controlador que principalmente se usa en drones ahora implementada para robots submarinos, el Pixhawk se encarga de controlar todos los movimientos del robot mediante los diferentes sensores integrados en el robot submarino de Blue Robotics.

El primer paso para ejecutar el proyecto de diseñar un sistema de navegación autónomo para estabilizar un robot submarino para capturar imágenes y reconocimiento de especie marina “Estrella de Mar”, es el análisis de los distintos componentes electrónicos del Robot submarino de BlueRobotics, como se encuentra constituido el robot submarino, saber el funcionamiento de cada elemento, que protocolo de comunicación utiliza, si

utiliza algún tipo de software complementario o con qué tipo de software es compatible y comprender el correcto funcionamiento del controlador Pixhawk.

Otro punto importante es la implementación de la visión artificial al robot submarino utilizando la Raspberry Pi, la cual se comunicará por medio de puerto serial con la Pixhawk del robot. El algoritmo de procesamiento de imagen se la realizara en la Raspberry Pi y la librería OpenCv que es una librería orientada al procesamiento de imágenes en diferentes tipos de lenguaje de programación.

En este proyecto se busca dejar una base de conocimiento sobre la navegación autónoma de robots submarino utilizando el controlador Pixhawk que en la actualidad se está implementando en robots submarinos para una mayor estabilidad del robot, de igual manera contribuir con el avance de procesamiento de imágenes utilizando Raspberry Pi y OpenCv, donde se espera que este proyecto aporte a la comunidad estudiantil en el área de electrónica, automatización y telecomunicaciones de la Universidad Estatal Península de Santa Elena, e incentivar a que se interesen por el desarrollo tecnológico de la provincia.

## **2.2. Marco Conceptual**

En este capítulo se describen las definiciones de las herramientas necesarias de navegación autónoma y visión artificial para el desarrollo del proyecto.

### **2.2.1. Medio Acuático**

Como en este proyecto se utiliza un robot submarino que funcionara en un medio acuático es necesario conocer a que tipos de fenómenos físicos estará sometido que influirán en su desempeño.

#### **2.2.1.1. Flotabilidad**

Cuando un objeto es introducido parcial o totalmente al agua, de acuerdo con el principio de Arquímedes, este flotara siempre y cuando el peso del volumen de agua desplazada sea igual a su propio peso. Cuando un objeto flota es porque tiene flotabilidad positiva, pero si un objeto desplaza un volumen de agua de menor peso que el peso del objeto, este se hundirá, por lo que tendrá una flotabilidad negativa.

Un estado ideal para el funcionamiento del robot submarino es la flotabilidad neutra, que nos dice que, si el peso del volumen de agua desplazada es igual al peso del objeto, este no flotará ni se hundirá. Con la flotabilidad neutra el vehículo será capaz de moverse con

libertad en diferentes direcciones y reduciendo el esfuerzo de los motores. Como la flotabilidad depende de la densidad del agua se buscará tener una flotabilidad ligeramente positiva para evitar que el robot submarino se hunda hasta el fondo del mar ya que la densidad del agua salada es aproximadamente de  $1,028 \text{ kg/L}$  siendo más densa que el agua dulce ( $0,958 \text{ kg/L}$ ), por las sales minerales que se encuentran disueltas en su composición hacen que cualquier objeto o cuerpo flote más en el agua salada que en agua dulce [3].

### **2.2.1.2. Presión del agua**

El agua es un fluido que ejerce presión a todo cuerpo que se sumerja en ella, la presión que el agua ejerce a un cuerpo depende a la profundidad y densidad del líquido. Para saber la presión a la que se somete un objeto o cuerpo se utiliza la siguiente formula:

$$P = \rho \cdot g \cdot h$$

Donde tenemos:

$\rho$ = es la densidad del fluido

$g$ = gravedad ( $9.8 \text{ m/s}^2$ )

$h$ = es la profundidad

Se debe calcular la presión absoluta debido a que se toma en cuenta la presión atmosférica en la superficie, se la calcula con la siguiente formula:

$$P = P_o + \rho \cdot g \cdot h$$

Donde:

$P_o$ = es la presión atmosférica

La presión atmosférica a nivel del mar se mantiene relativamente constante, utilizando un estándar como referencia de 1 atm (atmósfera) debido a la presión atmosférica, dentro de diez metros de agua salada se ejerce una presión de 1 atm, por lo tanto, la presión aumentará 1 atm cada 10 metros de profundidad [3].

### **2.2.1.3. Hidrodinámica**

La hidrodinámica es la ciencia que estudia un fluido en movimiento, para este proyecto será el agua. Debemos tener claro que, para el estudio de la hidrodinámica en el agua, la densidad del agua se mantendrá así se produzca cambio de presión y además el agua es



un fluido incomprensible que no tiene viscosidad, es decir que no se tomara en cuenta el esfuerzo constante producidos por la viscosidad o la fricción interna.

En el agua podemos tener flujo laminar o turbulento. Cuando las partículas de un fluido se mueven con trayectoria de línea recta, de manera ordenada y formando líneas paralelas tenemos que es un fluido laminar. Un fluido es turbulento cuando el fluido se mueve de manera desordenada formando círculos que por lo general pueden producir remolinos [3].

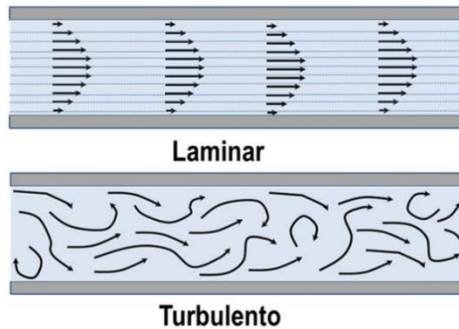


Figura 2: Flujo Laminar y Flujo Turbulento [11].

Como en este proyecto se trabaja con un robot submarino este estará expuesto a flujos turbulentos producidos por las corrientes marinas. Por lo que es necesario verificar que el robot se encuentre bien sellado para evitar filtraciones ya que estará sometido a un medio turbulento.

### 2.2.2. Robots Autónomos

Un concepto importante en este proyecto, es el concepto de robots autónomos. La definición más adaptada es la que hace referencia a robots no tripulados capaces de realizar tareas sin la ayuda del ser humano. Actualmente, estos tipos de robots no tripulados se encuentran en un constante avance tecnológico que despierta interés a profesionales e incluso a personas aficionadas.

Otra definición que se le puede dar a los robots autónomos es la que se menciona su funcionamiento general, el cual se basa en recopilar datos de su entorno a través de sensores, una vez que la información es recopilada se la procesa a través de la inteligencia artificial (IA) y por último se toma una decisión de acuerdo a la información antes procesada y se ejecuta una acción mediante los actuadores [12].

#### 2.2.2.1. Robot submarino

Antes de dar la definición del robot submarino, tenemos que tener en claro que es un submarino, se podría decir que la definición más adecuada, hace énfasis a maquinas que

realizan operaciones bajo el agua a distintas profundidades. La profundidad en que trabaja un submarino depende mucho del tipo de submarino y para que actividades entre en acción, ya que puede ser para fines de transporte, para actividades militares y para investigaciones científicas del fondo marino.

Una de las principales dudas que suele tener la mayoría de personas, es ¿Cómo se sumerge un submarino?, estos tienen compartimientos de aire que una vez se dé la orden de inmersión estos compartimientos se llenan de agua para que el submarino se sumerja, también se necesitara del movimiento de sus timones, otro factor a tener en cuenta es la fuerza de las propelas para que el submarino alcance la velocidad necesaria y que pueda sumergirse.

Recientemente los robots submarinos han revolucionado la exploración del fondo marino con fines investigativos, ofreciendo mayor cantidad de información extraída y a menos costo y sin que el ser humano corra algún tipo de peligro. Es por esto que podemos definir a los robots submarinos como vehículos no tripulados capaces de realizar diferentes tareas sin la intervención del hombre, destacándose su uso para investigación, mantenimiento y reparación a equipos bajo el agua y para monitorear el medioambiente acuático [13].

### 2.2.2.2. Clasificación de los robots submarinos

Se puede clasificar a los robots submarinos de acuerdo al nivel de autonomía, por el tipo de misión que realiza y por su sistema de propulsión.

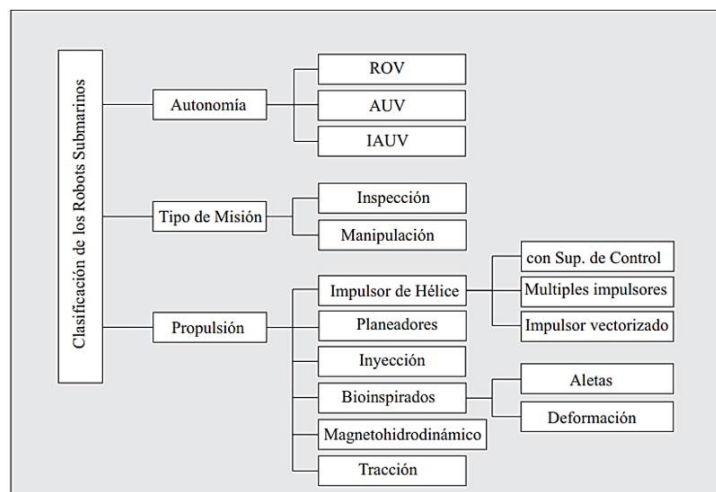


Figura 3: Clasificación de los robots submarinos [1].

En la figura 3 se muestra la clasificación de los robots submarino de manera detallada. La clasificación principal que se le da a los robots submarino es de acuerdo a al nivel de

autonomía siendo está en la que nos centraremos en este proyecto. En esta clasificación solo nos enfocaremos en los vehículos que deben ser controlados continuamente por un operador como los ROV (Vehículos Operados Remotamente), los que son completamente autónomos estos son los AUV (Vehículos Submarinos Autónomos) y por último tenemos a los IAUV que se los considera con un nivel intermedio de autonomía [1].

#### **2.2.2.2.1. Robots Submarinos Operados Remotamente ROV**

Conocidos como ROV por sus siglas en ingles Remotely Operated Vehicle son equipos que se encuentran conectados a la superficie por una correa (cable ethernet) que normalmente va conectado dos tarjetas Fathom-X una se encuentra a bordo del robot y la otra tarjeta se encuentra en la estación de control (ordenador) de la superficie, dicho cable permite el intercambio de datos y permite la ejecución de misiones autónomas desde la superficie [1].

#### **2.2.2.2.2. Robots Submarinos Autónomos AUV**

Los Vehículos Submarinos Autónomos más conocidos como AUV por las siglas en ingles Autonomus Underwater Vehicle, no poseen una correa (cable ethernet) de comunicación conectada a la superficie ya que se los programa para ejecutar misiones predefinida sin la supervisión de un operador. Cuando este tipo de vehículos desea enviar o intercambiar información con la superficie este los hace a través de dispositivos acústicos [1].

#### **2.2.2.2.3. Robots Submarinos Autónomos para Intervenciones IAUV**

Son vehículos considerados con nivel intermedio de autonomía, están diseñados para realizar misiones de manipulación. Gracias a este tipo de robots las misiones que requieran manipulación serían más económicas que los de tipo ROV, además al no contar con la correa de comunicación (cable ethernet) los hace perfecto para tener mejor maniobrabilidad [1].

#### **2.2.2.3. Robot Submarino BlueROV**

BlueRobotics es una compañía encargada de fabricar componentes para robótica marina de alto rendimiento y sobre todo a bajo costo, entre sus principales productos fabricados se encuentra el BlueROV.

El robot BlueROV, está compuesto con piezas de alta calidad, por una estructura resistente, seis propulsores T200 y baterías de reemplazo rápido, cuenta con potentes luces regulables para tener una excelente iluminación durante las misiones y poder tener transmisión de video de alta calidad en vivo. Al ser un robot de alta gama de bajo costo

en comparación con otros ROV lo hace asequible para la ejecución de misiones en aguas pocos profundas alcanzando hasta 100m bajo el mar [14].



Figura 4: Robot Submarino BlueROV [14].

### **2.2.3. Hardware Requerido**

En este apartado se mencionará los requerimientos de hardware para el desarrollo de la navegación autónoma y la visión artificial

#### **2.2.3.1. Kit BlueROV**

BlueRobotics proporciona un kit completo para el montaje del robot submarino, para realizar el montaje solo necesitan herramientas básicas ya que el kit viene incluido con herramientas para que el montaje sea de la manera más fácil posible. El kit BlueROV contiene los siguientes componentes:

1. 3 propulsores T200 en sentido horario y 3 en sentido antihorario
2. Paneles laterales e inferiores
3. Paneles centrales
4. Cradles de 4"
5. Abrazadera de 3"
6. Tarjeta Fathom-X (FXTI) incluye correa para ROV
7. Contrapesos
8. Carenados de flotabilidad
9. Batería lipo
10. Gabinete para la electrónica

11. Bolsas desecantes
12. Esponjas para sensores de fuga, probador de los sensores de fuga, adaptador de tarjeta SD, pegatinas y cierres.
13. Bomba de vacío, mangueras, accesorio de vacío y manual de uso.
14. Brindas grandes y dedal de sujeción.



Figura 5: Contenido del Kit BlueROV [15].

Hay que mencionar los principales componentes del kit que no fueron mencionados en el listado anterior como: Pixhawk incluye tarjeta SD, Raspberry Pi incluye tarjeta SD, tarjeta Fathom-X a bordo del robot, luz submarina y cámara USB [15].

### 2.2.3.2. Cámara

Los softwares ArduSub Companion y BlueOS proporcionan una serie de controladores que permiten la transmisión de video desde el vehículo a la estación de control de la superficie. El software ArduSub solo permite conectar una cámara a la vez mientras que el BlueOS admite conectar dos cámaras en su interfaz web, pero no se puede realizar múltiples transmisiones solo permite una a la vez.

#### 2.2.3.2.1. Cámaras USB

Las cámaras USB en su gran mayoría son compatibles con el software ArduSub y BlueOS, las especificaciones que deben tener son las siguientes:

- Salida H.264
- Resolución 10800 o 4K

La cámara USB probada y recomendada por BlueRobotics es:

### 2.2.3.2.1.1. Cámara USB Low-Light HD

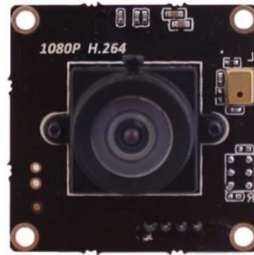


Figura 6: Cámara Low-Light HD [15].

Las especificaciones de esta cámara la hacen ideal para ser usada bajo el agua, proporciona un excelente rendimiento para condiciones de poca luz con un lente específico para brindar alta calidad de video en los robots submarinos, está basada en el sensor IMX322/323, el sensor de tamaño de imagen es de (1/2,9”) con un numero de pixeles bajo (2MP, 1080p), también cuenta con un chip de compresión H.264 para que todo el proceso de compresión de video sea interno. La placa de circuito interno de la cámara mide 32x32 mm e incluye cable USB de 8” y conector JST-PH [16].

**Tabla 1. Especificaciones de cámara USB Low-Light HD**

| PARÁMETRO                  | VALOR                          |
|----------------------------|--------------------------------|
| <b>LENTE</b>               |                                |
| Campo de visión Horizontal | 80°                            |
| Campo de visión vertical   | 64°                            |
| Longitud focal             | 2.97 milímetros                |
| <b>SENSOR</b>              |                                |
| Tipo                       | Sony Exmor IMX322/323          |
| Formato                    | 1/2.9”                         |
| Pixeles totales            | 2000(Alto) x 1121(V) – 2,24 MP |
| Iluminación mínima         | 0,01 lux                       |
| <b>VIDEO EN DIRECTO</b>    |                                |
| Estándar (min. – máx.)     | 240p15 – 1080p30               |
| Formato de compresión      | H.264/MJPEG/YUV2               |

### 2.2.3.2.2. Cámaras con interfaz serie de cámara (CSI)

Este tipo de cámara también compatible con el software ArduSub y BlueOS, fue una de las primeras cámaras que se utilizó en BlueROV hasta el 2017 que fue reemplazado por la cámara USB HD Low-Light HD por sus buenas condiciones bajo poca luz.

#### 2.2.3.2.2.1. Cámara Raspberry Pi 2

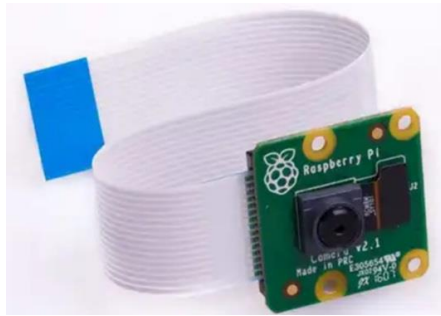


Figura 7: Cámara Raspberry Pi 2 [17].

En los inicios de BlueROV este tipo de cámara fue muy utilizado por tener buenas condiciones de poca luz, cuenta con el sensor IMX219 de 8 MP, con modos de video 720p60, VGA90 y 1080p30, además de realizar capturas de imágenes fijas. Posee un cable plano de 15cm el cual se conecta al puerto CSI de la Raspberry Pi [17].

### 2.2.3.3. Control electrónico de velocidad (ESC)

BlueRobotics generalmente para robótica submarina utiliza controles electrónicos de velocidad sin escobillas por lo que es necesario utilizar propulsores sin escobillas, ya que no se admite el uso de propulsores con y sin escobilla al mismo tiempo.

Los requisitos mínimos para cualquier tipo de control electrónico de velocidad (ESC) son:

- Control bidireccional
- Controlado por una entrada PWM
  - 1900 completamente hacia adelante
  - 1500 detenido
  - 1100 completamente hacia atrás

### 2.2.3.3.1. ESC sin escobillas

al utilizar un BlueROV para este proyecto nos centraremos en un ESC sin escobillas, el ESC probado y recomendado que es compatible con ArduSub y BlueOS es Basic ESC



Figura 8: Basic ESC - Controlador electrónico de velocidad [17].

Es un ESC (control electrónico de velocidad) sencillo bidireccional de 30 amperios, pero necesario para que funcione cualquier motor trifásico sin escobillas como el propulsor sin escobillas T200 que utiliza BlueRobotics. Posee rotación de dirección hacia adelante y hacia atrás para empuje y cuenta con bajo calor para situaciones de enfriamiento mínimo [18].

### 2.2.3.4. Propulsores

Para poder maniobrar un vehículo submarino son necesarios los propulsores que consta de un motor con o sin escobillas. El número de grados de libertad que tendrá un robot submarino va a depender del número y la orientación de los propulsores. Un factor a tener en cuenta es el consumo máximo de corriente que proporcione la fuente de alimentación ya que al consumo máximo de corriente del voltaje previsto se debe sumar para todos los propulsores [18].

#### 2.2.3.4.1. Propulsores sin escobillas

Este tipo de propulsores son ideales para robótica submarina ya que al no contar con escobillas no se tendrá inconvenientes con la protección de las escobillas y estar pendiente de su desgaste. BlueRobotics recomienda y ha probado el propulsor sin escobilla T200 por ser un motor inundado brinda gran potencia.



### 2.2.3.4.1.1. Propulsor T200



Figura 9: Propulsor T200 [14].

Tuvo sus inicios en el 2014 desde entonces se ha convertido en el propulsor submarino más utilizado para vehículos ROV y AUV, este propulsor consta de un motor sin escobillas inundado con devanados, un estator encapsulado y con imanes y rotor recubiertos. Sus hélices son de plástico de policarbonato, los metales expuestos al agua están hechos por acero inoxidable de 316 grado marino. El motor se enfría con agua y lubrica con agua los casquillos de plástico para su correcto funcionamiento necesita de un controlador electrónico de velocidad sin escobillas. Puede funcionar en variedad de voltaje dependiendo del rendimiento que el usuario desee, en la siguiente tabla se detalla a que voltaje puede funcionar [14].

**Tabla 2. Especificaciones técnicas de propulsores T200**

| PARÁMETRO  | VALOR                    |
|--|--------------------------|
| <b>Rendimiento</b>                                   |                          |
| Empuje FWD/REV con aceleración máxima a 12 V         | 3,71 / 2,92 kg           |
| Empuje FWD/REV con aceleración máxima a nominal 16 V | 5,25 / 4,1 kg            |
| Empuje FWD/REV con aceleración máxima a máximo 20 V  | 6,75 / 5,05 kg           |
| Empuje mínimo  | 0,02 kg                  |
| <b>Eléctrico</b>                                     |                          |
| Tensión de funcionamiento                            | 7-20 voltios             |
| Corriente de aceleración máxima a 12 V               | 17 amperios – 205 vatios |
| Corriente de aceleración máxima a nominal 16 V       | 24 amperios – 390 vatios |
| Corriente de aceleración máxima a máximo 20 V        | 32 amperios – 645 vatios |

#### 2.2.3.4.2. Propulsores cepillados

Los propulsores con escobillas deben contar con sellos de eje o deben ser sellados internamente y contar con un sistema de compensación de aceite. Por lo general son un poco más económicos que los propulsores sin escobillas, para su correcto funcionamiento deben de utilizar un ESC (controlador electrónico de velocidad) con escobillas.

#### 2.2.3.5. Modulo Power



Figura 10: Modulo de Potencia [19].

El módulo de potencia se encarga de regular el voltaje de alimentación que se le suministra al controlador Pixhawk, también permite sensar corriente y voltaje de tal modo que se pueda configurar acciones por falla de batería, maneja baterías hasta 4 celdas y corrientes por debajo de 90 A, por medio del software de superficie Mission Planner o QGround Control se podrá visualizar los niveles de batería o el consumo del vehículo durante una misión [19].

#### 2.2.3.6. Fuente de alimentación

La fuente de alimentación es fundamental para cualquier tipo de proyecto ya que se debe alimentar a todos los componentes electrónicos que componen al vehículo submarino, además el suministro de energía debe ser estable y se debe regular los voltajes más altos a voltajes más bajos para poder alimentar componente que no requieren de tanto consumo energético, para esto es necesario la utilización de reguladores de voltaje. BlueRobotics recomienda utilizar las baterías de iones de litio de BlueRobotics, Turnigy Li-po y a través de Tether (alto voltaje), esta última no es muy tomada en cuenta por su costo muy elevado.

##### 2.2.3.6.1. Batería de iones de litio de BlueRobotics (14,8 V, 18 Ah)



Figura 11: Batería de iones de litio de BlueRobotics (14,8 V, 18 Ah) [14].

Está fabricada con celdas de iones de litio 18650 de alta calidad, diseñada específicamente para uso de vehículos de BlueRobotics ya sea el BlueROV, BlueROV2 y BlueBoat. Es una batería de 4 celdas con una capacidad nominal de 18 Ah, proporcionando hasta 4 horas de uso. Las celdas que componen la batería brindan un excelente rendimiento y posee una gran tolerancia al mal manejo accidental, además cuenta con aprobación CE (Conformité Européenne) y cumple con las normativas ROHS [14].

**Tabla 3. Especificaciones batería de iones de litio de BlueRobotics (14,8 V, 18 Ah)**

| PARÁMETRO                             | VALOR          |
|---------------------------------------|----------------|
| Voltaje nominal                       | 14,8V          |
| Capacidad nominal                     | 18Ah – 266,4WH |
| Configuración de celda                | 4S6P           |
| Consumo máximo de corriente continua  | 60A            |
| Consumo máximo de corriente de ráfaga | 132A           |
| Corriente de carga máxima             | 20A            |
| Voltaje mínimo seguro                 | 12V            |

#### 2.2.3.6.2. Turnigy LiPo (14,8 V, 10 Ah)



Figura 12: Batería Turnigy LiPo (14,8V, 10 Ah) [20].

Las baterías Turnigy son conocidas por su gran rendimiento y sobre todo por su bajo costo, a pesar de que son más económicas que las baterías de BlueRobotics aguantan misiones de alto rendimiento de cualquier tipo de vehículo, brindan un tiempo de vuelo más largo con respecto a otras baterías LiPoly estándar, no aumentan el peso del vehículo y son ideales para drones multirrotor [20].

**Tabla 4. Especificaciones de batería Turnigy LiPo (14,8 V, 10 Ah)**

| PARÁMETROS                    | VALOR                    |
|-------------------------------|--------------------------|
| Capacidad mínima              | 10000 mAh                |
| Configuración                 | 4S1P / 14,8 V / 4 celdas |
| Descarga constante            | 12 C                     |
| Descarga máxima (10 segundos) | 24 C                     |
| Enchufe de carga              | 902g                     |
| Enchufe de descarga           | 168x69x38 mm             |
| Peso                          | JST-XH                   |
| Tamaño                        | XT90                     |

### 2.2.3.7. Regulador de Voltaje

Al usar una fuente de alimentación de hasta 14,8 V, 18 Ah, es necesario reducir el voltaje para aquellos componentes electrónicos que funcionan a partir de voltajes pequeños. El regulador de voltaje nos ayudara con esta tarea, el regulador de voltaje probado y recomendado por BlueRobotics es la fuente de alimentación de 5V 6A.

#### 2.2.3.7.1. Fuente de alimentación de 5 V 6 A



Figura 13: Fuente de alimentación de 5V 6A [18].

Esta fuente de alimentación acepta voltajes de entrada de 7V hasta 26V, proporciona una alimentación de 5V hasta 6A además utiliza un convertidor de computación con mínima pérdida de calor. Posee dos conectores estándar modo servo para facilitar la alimentación a dispositivos de bajo consumo (5V) como el controlador de vuelo Pixhawk, Raspberry Pi [18].

### 2.2.3.8. Pixhawk

El controlador de vuelo Pixhawk es una placa de hardware libre diseñado con el objetivo de proporcionar un sistema de piloto automático a distintas comunidades ya sea académica o industrial a bajo costo. El sistema de piloto automático que ofrece el controlador Pixhawk es una solución completa para vehículos autónomos ya que está basado en código abierto, cuenta con algoritmos para control de altitud y posicionamiento, también provee algoritmos para guía, navegación y control para distintos tipos de vehículos autónomos [21].

A pesar de ser un controlador de hardware libre, el Pixhawk fue diseñado por el proyecto PX4 y manufacturada por la empresa 3DR desde el año 2014, el objetivo de los diseñadores del Pixhawk era crear una plataforma multihilo y que esta sea capaz de dar un aumento el rendimiento de cualquier tipo de vehículo autónomo proporcionando flexibilidad y confiabilidad a los distintos usuarios.

A continuación se muestra en la siguiente tabla las características principales de la versión de Pixhawk PX4 que se usa en este proyecto:

**Tabla 5. Características principales del Pixhawk**

|                   |   |
|-------------------|---|
| <b>Procesador</b> | <ul style="list-style-type: none"><li>• ARM Cortex-M4 de 32 bits con FPU.</li><li>• Frecuencia de 168 MHz, 256 KB RAM y 2 MB de memoria Flash.</li><li>• Procesador a prueba de fallos ARM Cortex-M3 de 32 bits y frecuencia de 24 MHz.</li></ul>   |
| <b>Sensores</b>   | <ul style="list-style-type: none"><li>• MPU 6000 giroscopio y acelerómetro principal.</li><li>• Giroscopio ST Micro de 16 bits.</li><li>• Brújula/Acelerómetro ST Micro de 14 bits.</li><li>• Barómetro Medio (calibración configurable).</li></ul> |
| <b>Interfaz</b>   | <ul style="list-style-type: none"><li>• 5 puertos serial UART.</li><li>• Señal de suma PPM.</li><li>• Entrada RSSI 14 PWM.</li><li>• I2C, 2 CAN, SP1 y USB.</li><li>• SPKT/DSM entrada satélite.</li><li>• Entradas ADC de 3.3V y 6.6V.</li></ul>   |

|                    |  |
|--------------------|--|
| <b>Dimensiones</b> | <ul style="list-style-type: none"> <li>• Ancho: 50 mm 2,0”</li> <li>• Altura: 15,5 mm 0,6”</li> <li>• Longitud: 81.5 mm 3,2”</li> <li>• Peso: 38 gramos</li> </ul> |
|--------------------|--|

Cabe mencionar que también cuenta con pines de entrada y salida (analógicos y digitales), sistema operativo NuttX (Tiempo real), incluye bocina, botón de seguridad, tarjeta SD y Leds indicadores [22].



Figura 14: Pixhawk Puertos Frontales [22].



Figura 15: Pixhawk Puertos Laterales [22].

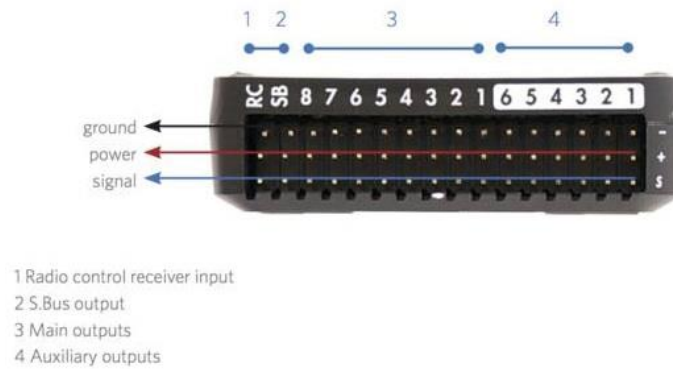


Figura 16: Pixhawk Pines de Entrada y Salida [22].

El sensor MPU 6000 es una de las características más importantes que posee el controlador Pixhawk, este sensor posee un giroscopio y un acelerómetro en los 3 ejes. El giroscopio tiene salida de datos de 16 bits en un rango de 250 a 2000 Grad/Seg. mientras que el acelerómetro también tiene salida de 16 bits en un rango de 2 a 16g. El Pixhawk también tiene integrado un barómetro MEAS MS5611 que tiene una salida de 24 bits en un rango de medición de presión de 10 a 1200 mBar. Además de que el Pixhawk cuenta con un procesador de respaldo ARM Cortex-M3 de 32 bits incluido un giroscopio de 16 bits y un acelerómetro de 14 bits [23].

De las 14 salidas de PWM 8 están destinadas para controlar los motores o ESC (Controladores de Velocidad Electrónicos), las cuales se pueden calibrar a conveniencia del usuario, las restantes salidas son denominadas auxiliares que pueden ser utilizadas para servomotores o disparador de una cámara.

Un aspecto importante del Pixhawk son los puestos incluidos cubriendo de esta manera el requisito mínimo de puertos además de que estos puertos detectan de manera automática los módulos conectados y los configura por defecto del fabricante siendo compatibles con los módulos detallados anteriormente, una ventaja a destacar es la configuración de los módulos las cuales se pueden modificar a conveniencia con ayuda del software de la estación terrestre ya sea QGround Control o Mission Planner donde se puede realizar la calibración de los sensores internos y externos del Pixhawk.

### 2.2.3.9. GPS

El módulo GPS nos permite conocer la posición de un objeto, es decir nos proporciona datos de posicionamiento de un vehículo. Se utilizan en los vehículos para que apoyen la misión de un ROV o AUV una vez que el submarino emerge a la superficie del mar para

conocer su posición, no se suele utilizar en vehículos sumergibles por pérdida de señal [1]. Este módulo GPS va conectado al puerto GPS del Pixhawk por medio del cable DF13 de seis posiciones y al puerto I2C con el cable DF13 de cuatro posiciones.

### **2.2.3.10. Raspberry Pi**

Para este proyecto utilizaremos la Raspberry Pi como computadora complementaria con el fin de que se conecte a la placa de piloto automático Pixhawk y se comunique por medio del protocolo MAVLink. El sistema operativo que se debe instalar en la Raspberry Pi deber ser el ArduSub Companion o BlueOS para que cumpla las funciones de transmitir video a la computadora de superficie y retransmite la comunicación entre la Pixhawk y la computadora de superficie a través de comunicación ethernet en este caso será la correa Fathom. Un dato muy importante a tener en cuenta es que el sistema operativo ArduSub Companion solo es compatible con la Raspberry Pi 3 modelo B, mientras que el BlueOS es compatible con la Raspberry Pi 3 modelo B y la Raspberry Pi 4 modelo B.

#### **2.2.3.10.1. Raspberry Pi 3 modelo B**



Figura 17: Raspberry Pi 3 modelo B [24].

Es el primer modelo de Raspberry Pi de tercera generación salió al mercado en febrero del 2016 cuenta con procesador de 64 bits de 4 núcleos a 1,2 GHz y con 1 GB de RAM [24]. Las características principales la Raspberry Pi 3 modelo B son:

- CPU Broadcom BCM2837 de 64 bits de 4 núcleos a 1,2 GHz
- 1 GB de RAM
- Bluetooth 4.1 y Wi-Fi 802.11 b/g/n
- GPIO extendido de 40 pines de entrada – salida
- Puerto HDMI de tamaño completo



- Puerto CSI para cámara Raspberry Pi
- Puerto DSI para pantalla táctil Raspberry Pi
- Puerto Micro USB para alimentación
- Ranura Micro SD para sistema operativo

### 2.2.3.10.2. Raspberry Pi 4 modelo B

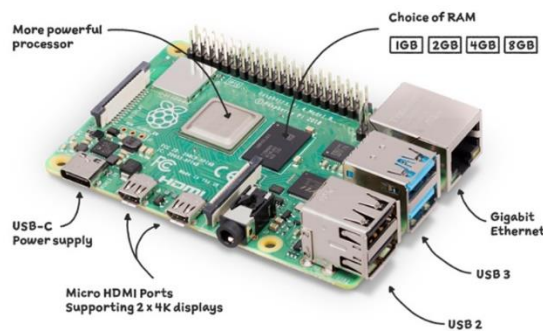


Figura 18: Raspberry Pi 4 modelo B [25].

La Raspberry pi 4 modelo B salió al mercado en el 2019, cuenta con un procesador ARM Cortex-A72 con 4 núcleos a 1,5 GHz, el tamaño de RAM varía de 1GB, 2GB, 4GB y 8GB dependiendo de los requerimientos del usuario, además permite la decodificación de videos en 4K a 60fps [25]. Entre las características principales de la Raspberry pi 4 modelo B tenemos:

- CPU Broadcom BCM2711, Cortex-A72 de 4 núcleos (ARM v8) de 64 bits
- RAM de 1GB, 2GB, 4GB y 8GB LPDDR4-3200 (depende del modelo)
- Wi-Fi IEEE 802.11ac de 2,4 GHz y 5 GHz y bluetooth 5.0
- Gigabit Ethernet (RJ45)
- 40 pines de entrada-salida digitales GPIO
- 2 puertos USB 3.0 y 2 puertos USB 2.0
- Puerto de cámara MIPI CSI
- Puerto de pantalla DSI de 2 carriles para pantalla táctil
- 2 puertos Micro HDMI admite hasta 4kp60

- H.264 decodificación de video a 1080p60 y H.265 decodificación de video a 4kp60
- Ranura Micro SD para cargar el sistema operativo
- 5V DC vía conector USB-C
- 5V DC vía conector GPIO

### 2.2.3.11. Fathom-X

La placa Fathom-X facilita una conexión ethernet de gran velocidad a larga distancia para el BlueROV. La placa se conecta al BlueROV por medio de la correa Fathom que es un cable Cat5 estándar, también se puede utilizar un solo par de cables trenzados, además utiliza el módulo HomePlug y cumple con el estándar IEEE-1901 [14]. Se utilizan dos placas Fathom-X para poder conectar el BlueROV con la estación de control terrestre, una placa va a bordo del vehículo y la otra se encuentran en la superficie.



Figura 19: Placa Fathom-X que va a bordo del BlueROV [14].

#### 2.2.3.11.1. Fathom-X Tether Interface (FXTI)



Figura 20: Placa Fathom-X (FXTI) [14].

Esta placa es la que se encuentra en la estación de control terrestre donde se conecta la correa Fathom del BlueROV, es un gabinete simple y expandible que permite la

interacción con el BlueROV y de otros dispositivos que usan la comunicación HomePlug. La Fathom-X Tether Interface cuenta con conexión USB por donde se proporciona energía a la caja e internamente está conectada a un adaptador USB a ethernet que va conectado a una placa Fathom-X. el conector Binder 770 que posee FXTI permite una conexión limpia desde la computadora con la correa Fathom, permitiendo de esta manera la conexión del BlueROV con el software que se utilice para el monitoreo del robot [14].

#### **2.2.3.12. Correa Fathom**



Figura 21: Correa Fathom [18].

La correa Fathom es un cable de diámetro de 4 mm y 7,6 mm de alta calidad fabricado específicamente para BlueROV, cuenta con flotabilidad neutra, posee resistencia a la rotura de 350 lb, esta incrustado con fibras que bloquean el agua para evitar cualquier tipo de filtración. El interior de la correa contiene hilos de kevlar para mayor resistencia y para llenar espacio contiene fibras de Dacron, además estas fibras se encuentran impregnadas con un compuesto bloqueador el agua para evitar cualquier tipo de fuga causa por rasgaduras en el cable. La correa de espuma de poliuretano amarilla es resistente a la abrasión, es lo suficientemente flexible para permitir que el BlueROV se pueda mover con libertad [18].

#### **2.2.4. Software Requerido**

En este apartado se mencionará los requerimientos de software para el desarrollo de la navegación autónoma y la visión artificial.

##### **2.2.4.1. Software complementario (ArduSub Companion o BlueOS)**

El software complementario se instala en la placa Raspberry pi, existen varios tipos de software que son compatibles con esta placa, pero para robots submarinos desde el 2015 se empezó a utilizar el software ArduSub Companion, en sus inicios Companion fue

creado solo para poder enrutar la transmisión de video y la comunicación de robot submarino con la computadora de superficie [8].

#### 2.2.4.1.1. ArduSub Companion

La imagen del sistema operativo se lo puede descargar desde el sitio oficial de ArduSub hay que tener en cuenta que solo es compatible con la placa Raspberry Pi 3 modelo B. El software ArduSub Companion es una modificación del sistema operativo Raspbian, para que sea funcional se debe instalar la imagen en una tarjeta SD. la instalación se la hace mediante el programa **balenaEtcher**, donde debemos seleccionar el software descargado y la tarjeta SD donde vamos a instalar, terminada la instalación se introduce la tarjeta SD en la placa Raspberry pi [8].

#### 2.2.4.1.2. BlueOS

BlueOS es el reemplazo del software ArduSub Companion, una de sus principales novedades es la compatibilidad con Raspberry Pi 3 y con Raspberry Pi 4, recordemos que Companion solo era compatible con Raspberry Pi 3 modelo B. cabe mencionar que este software se diseñó y creado desde cero con el fin de que se pueda adaptar a cualquier tipo de robot submarino.

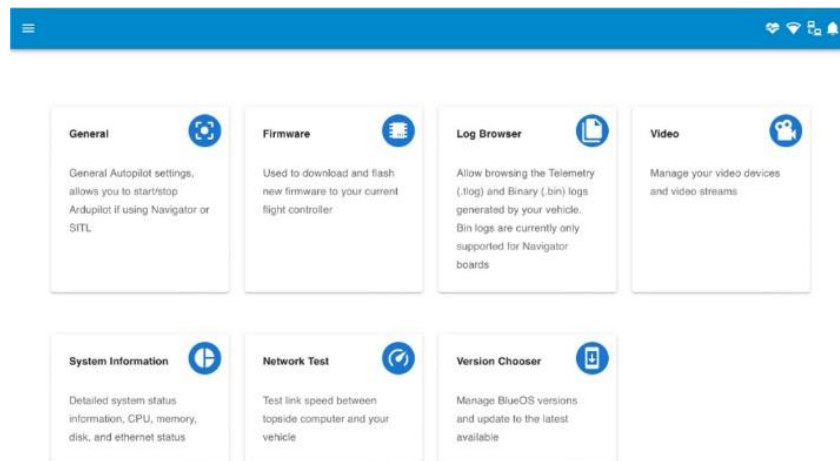


Figura 22: Interfaz Web de BlueOS [8].

En la figura 22 se puede apreciar la interfaz web con la que cuenta BlueOS, una interfaz amigable que brinda facilidad de manejo, donde tendrán acceso al sistema del robot, podrán acceder a la cámara de video y al enrutamiento del robot. El usuario podrá acceder a esta interfaz ingresando al navegador de su preferencia y escribiendo la siguiente dirección [192.168.2.2](http://192.168.2.2) o [blueos.local](http://blueos.local) al momento en que la interfaz detecte un vehículo podrá tener acceso y podrá cambiar lo que desee de acuerdo al uso que le vaya a dar [8].

#### **2.2.4.2. Firmware**

El Firmware a utilizar en este proyecto es de ArduPilot, es un software libre que ha estado en desarrollo desde el 2010 el cual se puede redistribuir y modificar por el usuario según los términos de la licencia GNU publicada por Free Software Fundación [26]. ArduPilot es considerado el software más avanzado de código abierto que proporciona piloto automático a cualquier sistema de vehículo autónomo, entre los firmwares estables que proporciona ArduPilot se encuentran los siguientes:

- Copter: para helicópteros y multicópteros o drones.
- Plane: para vehículos tipos avión.
- Rover: para vehículos terrestres y barcos.
- AntennaTracker: se usa para seguimiento de antenas de vehículos ArduPilot.
- Sub: para vehículos submarinos.

##### **2.2.4.2.1. Firmware Sub**

El firmware sub se originó a partir del código de ArduCopter, funciona correctamente para ROV (vehículos submarinos operados remotamente) y AUV (vehículos submarinos autónomos). Está diseñado para ser un firmware seguro y fácil de usar para los usuarios, también incluye control de rumbo, profundidad, estabilidad por retroalimentación y navegación autónoma.

En la actualidad este firmware está a la vanguardia de la robótica marina por su fácil manejo y al ser de código abierto es ideal para las modificaciones y pruebas que el usuario desee implementar. Además, funciona correctamente con el software QGround Control y Mission Planner donde se puede planificar misiones autónomas y monitorear la telemetría del vehículo.

##### **2.2.4.3. Protocolo de MAVLink**

Es un protocolo ligero diseñado específicamente para facilitar la comunicación entre un dron y la estación de control terrestre QGround Control o Mission Planner. El protocolo fue creado en el 2009 con la finalidad de proporcionar una comunicación sencilla, fiable y sobre todo ligera que sea capaz de poder integrarse a cualquier tipo de vehículo aéreo como su nombre mismo los dice MAVLink (Micro Air Vehicle Link) Enlace de vehículo Micro Aéreo [27].

Actualmente el protocolo MAVLink se integra fácilmente a cualquier tipo de vehículo y a diferente software que se use como estación de control terrestre. La trama de paquete de MAVLink se encuentra dividida en 8 campos, además se pueden crear nuevos mensajes personalizados [28].

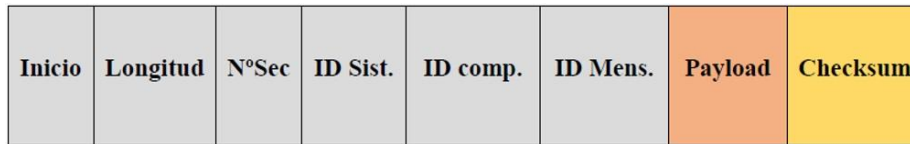


Figura 23: Trama de paquete del protocolo MAVLink [28].

Los primeros seis campos forman la cabecera de la trama de paquete de MAVLink y cada uno de estos 6 primeros campos corresponde con 1 byte:

- **Inicio:** Es el comienzo del paquete.
- **Longitud:** Esta comprendida con valores entre 0 a 255 los cuales indican el número de bytes que se envían.
- **Número de secuencia (Nº Sec.):** En este campo se muestra el número de paquetes enviados.
- **ID del Sistema (ID Sist.):** En este campo se identifica el sistema para poder diferenciar entre varios de ellos.
- **ID del Componente Emisor (ID Comp.):** Es el que permite diferenciar componentes dentro del mismo sistema.
- **ID del Mensaje (ID Mens.):** En este tramo se indica el contenido del mensaje y modo de lectura.
- **Payload:** Aquí es donde se almacena la información a transmitir.
- **Checksum:** Donde se controla los errores que pueden aparecer.

#### 2.2.4.4. Mission Planner



Figura 24: Interfaz de Mission Planner

Imagen elaborada por el autor

Mission Planner es un programa de estación terrestre creado por Michael Osborne para el proyecto de piloto automático bajo el software de ArduPilot de código abierto. Por medio de este programa se puede controlar distintos vehículos tales como aviones, multicopteros, vehículos terrestres (Rover), barcos y submarinos.

La interfaz de Mission Planner es bastante amigable y fácil de manejar para nuevos usuarios, también se la puede usar como una plataforma de configuración de distintos vehículos [23]. Dentro de las funciones que puede realizar Mission Planner tenemos las siguientes:

- Cargar el firmware al piloto automático.
- Configurar, calibrar y ajustar el vehículo automático para un mejor rendimiento.
- Crear, guardar y cargar misiones autónomas al piloto automático del vehículo.
- Descarga de registro de las misiones para análisis de las mismas.
- Monitoreo en tiempo real del vehículo durante una misión autónoma.
- Control manual del vehículo mediante un joystick.
- Utiliza Google Maps/Open Street Map/Bing para puntos de referencia.

### 2.2.4.5. QGround Control

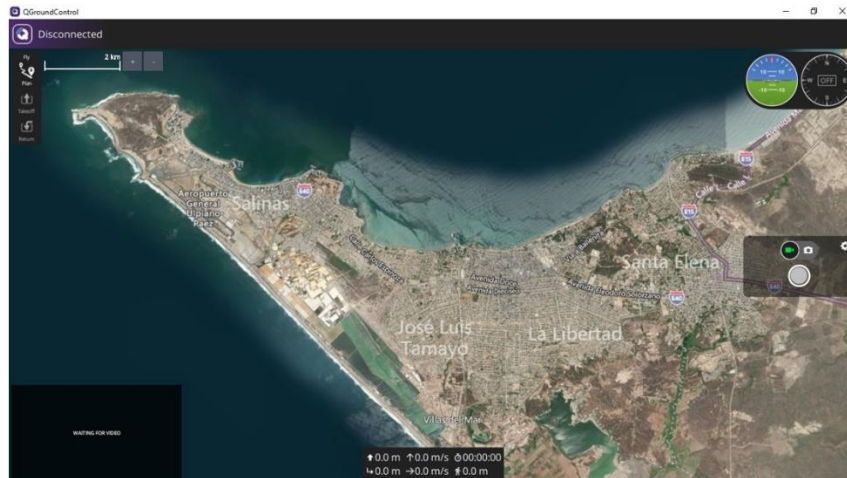


Figura 25: Interfaz de QGroundControl

Imagen elaborada por el autor

QGroundControl es un programa que brinda control total de vuelo y planificación de misiones autónomas para cualquier vehículo como drones, barcos, vehículos terrestres y submarinos, trabaja muy bien con plataformas PX4, ArduPilot o cualquier otra controladora de vuelo. Es un programa que con una interfaz sencilla fácil de manejar incluso para principiantes [28]. Entre las funciones más importantes que ofrece QGroundControl tenemos las siguientes:

- Permite configurar, calibrar y ajustar el vehículo ya sea de la plataforma PX4, ArduPilot o cualquier otra controladora de vuelo.
- Crear, guardar y cargar misiones autónomas.
- Monitorización de datos y mapeo de la trayectoria de la misión.
- Permite gestionar varios vehículos.
- Graba y guarda videos de la misión.
- Compatible para móvil con sistemas Android o iOS, para PC con Windows, OS X y Linux.

### 2.2.4.6. Python

Python es un lenguaje de programación creado por Guido Van Rossum en 1989, es considerado un lenguaje programación de alto nivel, dinámico, multiplataforma, multiparadigma y fácil de usar. Al ser un lenguaje interpretado no requiere de un



compilador sino de un intérprete para poder ejecutarse. Al ser independiente de arquitectura hardware dispone libre grado de portabilidad y es un lenguaje que acepta diferentes métodos de programación de acuerdo para al uso de aplicación, además es un lenguaje multiplataforma que puede ser ejecutado en diferentes sistemas operativos [28].

### **2.2.5. Visión artificial**

La visión por computadora o visión artificial es un campo de la inteligencia artificial la cual le permite a las computadoras ver y extraer información por medio de videos, imágenes digitales y otras entradas visuales, la información captada por estos medios le permiten al ordenador procesar la información y en base a ellos tomar medidas de acuerdo al caso aplicado.

La visión artificial trata de imitar la visión humana, pero para que esto sea posible se debe otorgar a las máquinas el poder de la visión, es decir las maquinas necesitan adquirir, procesar, analizar y comprender imágenes [29]. Esto es posible gracias a las redes neuronales que usa el proceso de aprendizaje iterativo, este proceso comienza proporcionándole a la maquina un conjunto de datos que ayuda a que esta pueda aprender un tema determinado. la ventaja que tienes las maquinas es que pueden procesar y analizar miles de datos en poco tiempo que puede superar fácilmente a las capacidades humanas [30].

#### **2.2.5.1. Detección de Objetos**

La detección de objetos es una técnica que le permite a la maquina localizar uno o varios objetos dentro de imágenes o videos. Para obtener mejores resultados la detección de objeto aprovecha las técnicas de deep learning (aprendizaje profundo) o machine learning (aprendizaje automático) con estas técnicas la detección de objeto busca replicar la manera en que el humano reconoce y localiza objetos [31].

Se puede empezar con la detección de objetos por medio del aprendizaje profundo donde se utilizan CNN (red neuronal convolucional) o YOLO que son técnicas que aprenden a detectar objetos dentro de imágenes. Se puede realizar un detector de objetos personalizado, pero para eso es necesario diseñar desde cero una arquitectura de red para poder identificar las características de los objetos que son de interés. También se puede utilizar detección de objeto personalizada con redes previamente entrenadas y ajustarla a las necesidades que requiera el usuario, este método es el más utilizado ya que proporciona resultados más rápidos y con resultados eficientes [32].

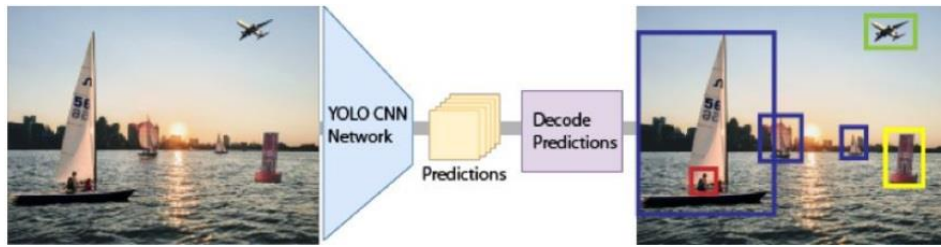


Figura 26: Descripción general de la detección de objetos de YOLO [32].

### 2.2.5.2. Red neuronal

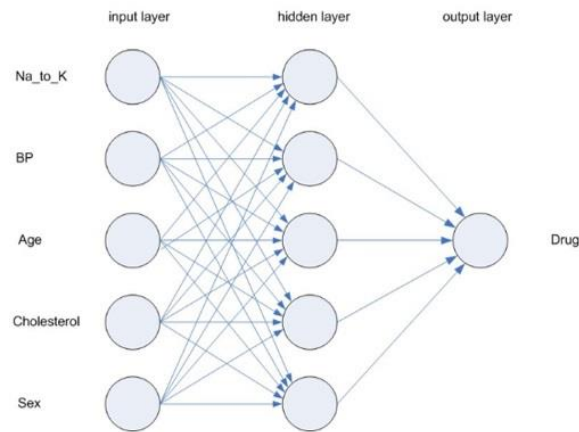


Figura 27: Estructura de una Red Neuronal [33].

Una red neuronal es un modelo simple que enseña a las computadoras a procesar los datos de la forma en que el cerebro humano procesa la información. Este proceso funciona sincronizando un gran número de unidades de procesamiento conectados entre sí tratando de emular a las neuronas del ser humano, es decir se utilizan nodos o neuronas interconectadas organizadas en capas. Por lo general una red neuronal está conformada por 3 capas: la primera es la capa de entrada; en la capa oculta pueden ir varias capas dependiendo del tipo de aprendizaje; la última es la capa de salida donde muestra los aprendidos en el proceso. Mediante este proceso las computadoras aprenden examinando los registros individuales de cada capa, generando una predicción de cada registro y realiza el análisis de dicha predicción ya sea correcta o incorrecta esta predicción, para que el ordenador aprenda de manera eficiente es necesario repetir varias veces este proceso para que la red neuronal aprenda de los errores y mejore las predicciones hasta alcanzar un nivel alto de precisión [33].

### 2.2.5.3. Red neuronal convolucional

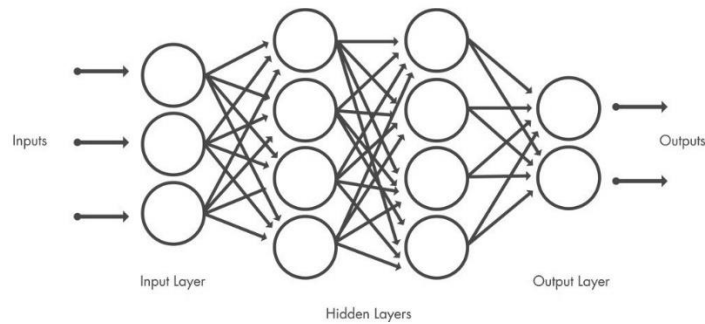


Figura 28: Estructura de una Red Neuronal Convolutiva [34].

Una red neuronal convolucional o CNN por sus siglas en inglés (convolutional neural networks) es una arquitectura de red que utiliza aprendizaje profundo y aprende directamente a partir de conjunto de datos.

Una CNN está compuesta por 3 capas al igual que una red neuronal con la diferencia que en la capa oculta se puede tener cientos de capas con el fin de detectar y analizar diferentes características de una misma imagen. A la capa oculta se le aplican filtros al conjunto de imágenes de entrenamiento con el fin de que se pueda analizar con distintas resoluciones, color, perspectiva, etc., la salida que se tenga de cada proceso se empleara como entrada para la siguiente capa. Este proceso se realiza para que la red neuronal convolucional obtenga características particulares que definen a un objeto de forma singular y sea más sencillo obtener resultados eficientes [34]. Dentro de la capa oculta existen 3 capas comunes que son:

- **Convolución:** En esta capa se aplican filtros convolucionales al conjunto de imágenes de entrada, los filtros aplicados activan diferentes características de la imagen procesada.
- **Unidad lineal rectificadora (RELU):** Con el fin de obtener un entrenamiento más rápido y eficaz esta capa mantiene los valores positivos y establece en cero los valores negativos, en otras palabras, esta capa solo prosigue a la siguiente capa las características activadas de la imagen.
- **Agrupación:** Se encarga de simplificar la salida reduciendo la tasa de muestreo disminuyendo el número de parámetros que la red tiene que aprender.

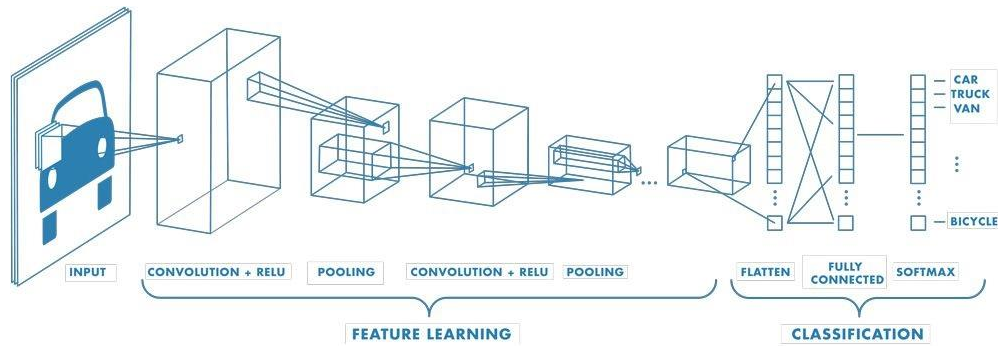


Figura 29: Ejemplo de estructura de una CNN [34].

Por medio de los pesos y valores de sesgos que posee una red neuronal convolucional, una red entrenada por con CNN es capaz de reconocer un determinado objeto en cualquier parte de la imagen en que aparezca dicho objeto ya que esto hacen que las capas ocultan detecten y analicen las mismas características como bordes o formas en diferentes regiones de la imagen que está siendo procesada.

#### 2.2.5.4. YOLO

YOLO (You Only Look Once), es un algoritmo de detección de objetos, destaca por realizar de forma rápida el proceso de detección de objetos en tiempo real, introducido al mundo tecnológico en el 2015. Yolo utiliza la técnica Darknet, que es un tipo de red neuronal de código abierto escrito en C, que admite cálculo de CPU y GPU (mucho más rápido que CPU) [35].

- La arquitectura de YOLO posee un total de 24 capas convolucionales, donde 4 capas son de agrupamiento máximo y 2 capas se encuentran conectadas. A continuación, se detalla cómo funciona la arquitectura de YOLO.
- Una vez que se ingresa el conjunto de datos se cambia el tamaño de la imagen a 448x448 antes de pasar por la red convolucional.
- Para reducir el número de canales se aplica una convolución 1x1, después se aplica una convolución 3x3 para que genere una salida cuboidal
- La unidad lineal rectificadora más conocida como RELU se encarga de activar las características de la imagen, es decir los valores positivos se mantienen mientras que los ceros los convierte en cero, este proceso no se aplica en la capa final la cual utiliza una función de activación lineal.

- Para evitar ajustes en exceso de los parámetros que se desea que la red aprende se aplican técnicas adicionales como la normalización por lotes.

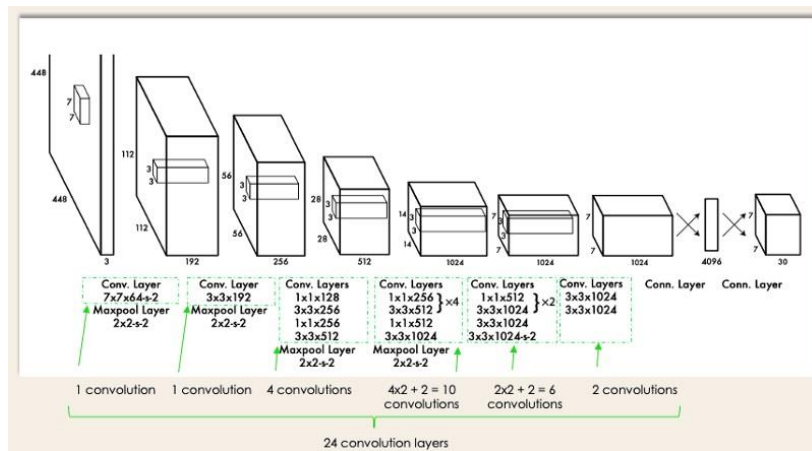


Figura 30: Arquitectura YOLO [36].

### 2.2.5.5. OpenCV

OpenCV (Open Source Computer Vision) es una librería de código abierto empleada para el procesamiento de imágenes en tiempo real. Es un software desarrollado por Intel en el año 1999, está escrito en lenguaje C y C++ y compatible con los sistemas operativos Windows, GNU/Linux, OS X, iOS y Android. Esta librería dispone de más de 2500 algoritmos y documentación de muestra para visión por computadora en tiempo real, los lenguajes de programación como Python, C++, Matlab, entre otros lenguajes de programación hacen que el uso de OpenCV sea más sencillo de manejar [37].

## CAPITULO III

### 3. Diseño del algoritmo de visión artificial

En este capítulo se explicará el procedimiento para diseñar un sistema de detección de objetos, para esto será necesario entrenar una red neuronal convolucional cuyo objetivo final es detectar una especie marina “Estrella de Mar”.

#### 3.1. Desarrollo de un modelo personalizado para detección de objetos

Existen una gran variedad de redes neuronales de detección de objetos preentrenada que se pueden utilizar como punto de partida para entrenar una red personalizada, es decir para que la red previamente entrenada aprenda una tarea nueva. El modelo preentrenado que vamos a utilizar en este proyecto es YOLOv4-Tiny, es un modelo de detección de objeto basado de YOLOv4, pero en una versión comprimida, posee una estructura de red simplificada con parámetros reducidos para ser aplicado a maquinas con menor potencia informática y dispositivos móviles [38].

##### 3.1.1. Elección del modelo YOLOv4-Tiny

La principal razón por la que se eligió trabajar con el modelo YOLOv4-Tiny es porque necesita de pocos requerimientos informáticos, lo que hace conveniente para que pueda ser implementada en la Raspberry Pi 4 y pueda ejecutar sin problemas el proceso de detección de objetos y que el mismo no presente retraso en tiempo real. Otra razón importante por la que se decidió trabajar con este modelo es que en la Raspberry Pi 4 aparte de ejecutar el proceso de detección de la Estrella de Mar se va a ejecutar el software BlueOS el cual ejecuta algunos scripts que son fundamentales para que el robot submarino trabaje correctamente. Antes de elegir este modelo se hicieron pruebas con el modelo YOLOv4 y YOLOv5, pero al ejecutar varios procesos a la vez producía sobrecalentamiento excesivo en la Raspberry Pi 4 y el sistema de detección presentaba retraso excesivo. Con el modelo YOLOv4-Tiny el proceso de detección ya no presentaba un retraso excesivo y se evitó el sobrecalentamiento excesivo [38].

En comparación con YOLOv4, YOLOv4-Tiny solo ha sido entrenado con 29 capas de convolución mientras que YOLOv4 posee 137 capas convolucionales. Con respecto a FPS (cuadros por segundos) YOLOv4-Tiny es ocho veces mayor que YOLOv4, pero decae en precisión ya que es de 2/3 de la de YOLOv4 [38].

La compensación entre la precisión y velocidad del modelo YOLOv4-Tiny es un aspecto a tener en cuenta dependiendo de su aplicación y los requerimientos computacionales que posee el equipo donde se ejecuta el modelo. A continuación, se explica esta compensación y si podría afectar a la identificación de la Estrella de Mar:

**Tamaño del modelo:** El tamaño del modelo es mucho más reducido ya que solo posee 29 capas convolucionales con aproximadamente 5.9 millones de parámetros por lo que hace que el modelo sea más ligero que YOLOv4 con 137 capas convolucionales con más de 64 millones de parámetros. La reducción de capas del modelo comprende menos exigencia en recursos de hardware permitiendo que sea aplicado a equipos que poseen capacidad limitada.

**Precisión:** Al contar con menos capas convolucionales provoca una simplificación de la arquitectura del modelo YOLOv4-Tiny lo que significaría que el modelo tiene menos capacidad para aprender características específicas del modelo entrenado produciendo de esta manera reducción en la precisión, es decir el modelo podría presentar mayor tasa de falsos positivos y falsos negativos lo que significa que el modelo podría no identificar algunos objetos o identificarlos erróneamente. Este problema se podría compensar con un conjunto de datos más amplio, mientras más imágenes con diferentes características se le proporcione al modelo este aprenderá a identificar con mayor exactitud las características del objeto entrenado.

**Velocidad:** El modelo YOLOv4-Tiny por su arquitectura simplificada posee significativamente mayor velocidad de detección llegando a alcanzar velocidades de hasta 443 FPS en hardware con un alto rendimiento computacional. Cuando se trata de detección de objetos en tiempo real la rapidez de predicción es más importante que la precisión. Esta es una de las razones por la que se eligió el modelo de YOLOv4-Tiny para entrenar un modelo de detección de objetos personalizado, además de tener un promedio de pérdida alrededor 0,15 después de 6000 [39].

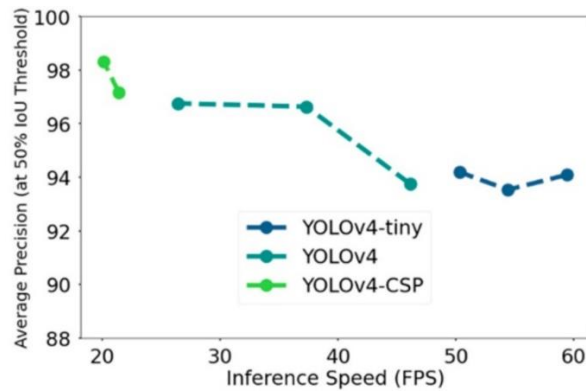


Figura 31: Grafica de comparación de modelos YOLOv4 en Precisión Promedio al 50% y velocidad de inferencia FPS [38].

### 3.1.2. Crear conjunto de datos

Para crear una red neuronal personalizada es necesario tener un buen conjunto de datos de imágenes es indispensable para poder entrenar el modelo personalizado, como mínimo es necesario tener unas 400 imágenes, además hay que tener en cuenta que el tamaño de las imágenes debe ser de acuerdo al modelo YOLOv4-Tiny (416x416) que se va a usar como base para entrenar red neuronal convolucional personalizada.

Podemos obtener gran cantidad de imágenes de la siguiente manera, descargando las imágenes de Google o por medio de codificación donde capturaremos imágenes del elemento que queremos entrenar. Si es por medio de Google, buscamos **Download All Images** y agregamos esa extensión a nuestro navegador. Ahora buscamos las imágenes que necesitaremos para entrenar la red neuronal, para este proyecto serán imágenes de Estrella de Mar, habilitamos la extensión de descarga y esta nos descargara un archivo zip con todas las imágenes de estrella de mar.

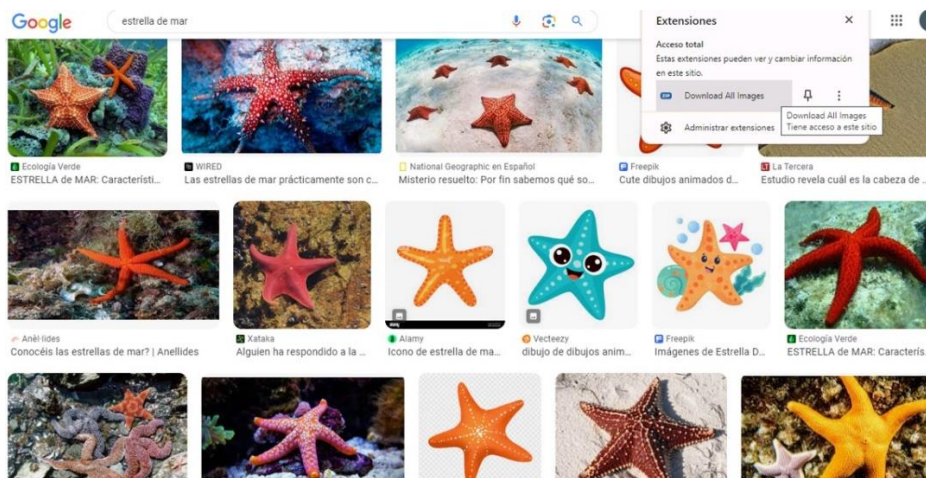
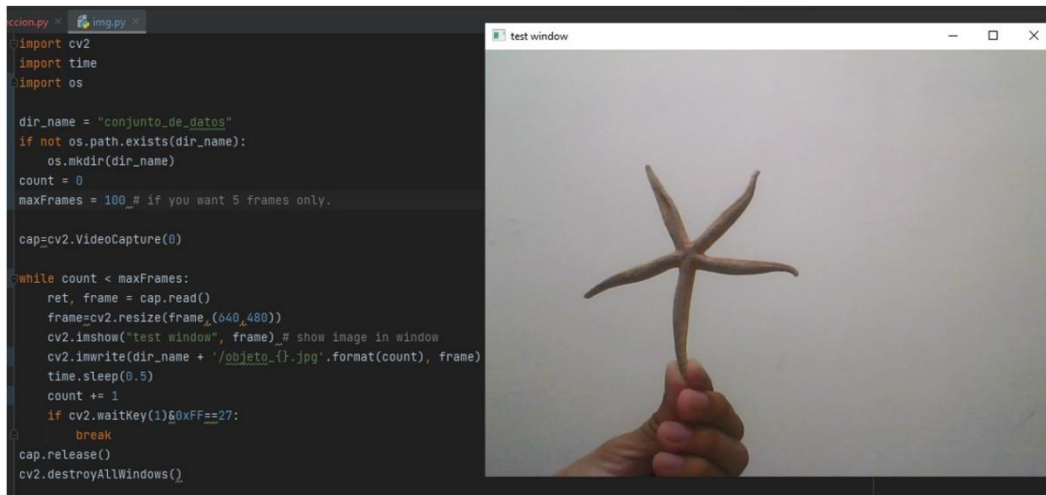


Figura 32: Descarga de conjunto de datos

Imagen elaborada por el autor



Para obtener el conjunto de datos de imágenes por medio de codificación es necesario establecer el número máximo de imágenes a guardar, de que tamaño se va a guardar la imagen, donde se va a guardar las imágenes y crear un bucle que me permita captar imágenes hasta el rango máximo establecido.



```
ccoon.py  img.py
:import cv2
:import time
:import os

dir_name = "conjunto_de_datos"
if not os.path.exists(dir_name):
    os.mkdir(dir_name)
count = 0
maxFrames = 100_# if you want 5 frames only.

cap=cv2.VideoCapture(0)

while count < maxFrames:
    ret, frame = cap.read()
    frame=cv2.resize(frame,(640,480))
    cv2.imshow("test window", frame)_# show image in window
    cv2.imwrite(dir_name + '/objeto_{}.jpg'.format(count), frame)
    time.sleep(0.5)
    count += 1
    if cv2.waitKey(1)&0xFF==27:
        break
cap.release()
cv2.destroyAllWindows()
```

Figura 33: Código en Python para crear conjunto de datos

Imagen elaborada por el autor

Como también se obtuvo imágenes desde Google para variar el contenido del conjunto de datos se descargaron imágenes con otras dimensiones a las requeridas por el entrenamiento de YOLOv4-Tiny, para esto es necesario dimensionar las imágenes al tamaño requerido 416x416. La herramienta **Image Resizer** de Google, nos ayudara a dimensionar las imágenes a 416x416 con buena calidad, una vez dentro de la página subimos las imágenes de nuestro conjunto de datos y colocamos las dimensiones que necesitamos, terminado el proceso descargamos el conjunto de imágenes ya dimensionadas.

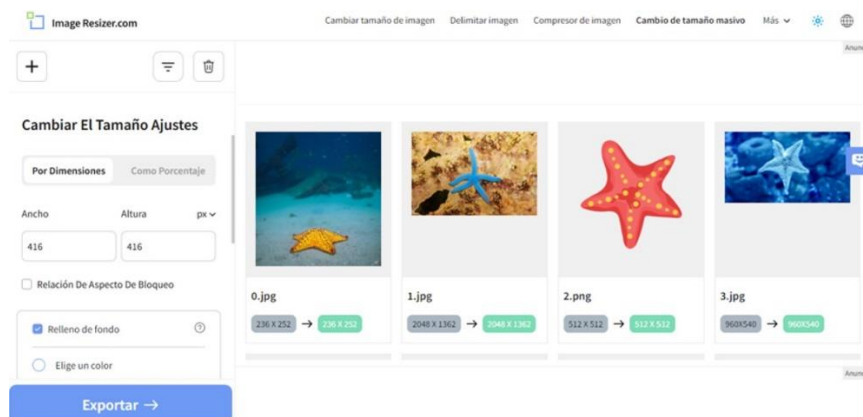


Figura 34: Imager Resizer - redimensión de imágenes

Imagen elaborada por el autor



```
Símbolo del sistema - pip install labelimg
Microsoft Windows [Versión 10.0.19045.2846]
(c) Microsoft Corporation. Todos los derechos reservados.

C:\Users\usuario>cd C:\Users\usuario\Documents\Deteccion Python

C:\Users\usuario\Documents\Deteccion Python>pip install labelimg
Collecting labelimg
  Using cached labelimg-1.8.6.tar.gz (247 kB)
Collecting pyqt5
  Using cached PyQt5-5.15.9-cp37-abi3-win_amd64.whl (6.8 MB)
Collecting lxml
  Downloading lxml-4.9.3-cp39-cp39-win_amd64.whl (3.9 MB)
    | 3.9 MB 547 kB/s
Collecting PyQt5-Qt5>=5.15.2
  Using cached PyQt5_Qt5-5.15.2-py3-none-win_amd64.whl (50.1 MB)
Collecting PyQt5-sip<13,>=12.11
  Downloading PyQt5_sip-12.12.1-cp39-cp39-win_amd64.whl (78 kB)
    | 78 kB 2.1 MB/s
Building wheels for collected packages: labelimg
  Building wheel for labelimg (setup.py) ... done
  Created wheel for labelimg: filename=labelimg-1.8.6-py2.py3-none-any.whl size=261588 sha256=5f5a704332051490a9ac6aa38d4545129ed0a33cbc7d3fd0db7cc0742a2a2095
  Stored in directory: c:\users\usuario\appdata\local\pip\cache\wheels\af\2d\29\af47d232f5b03ec3e64a8432032f328dfdea1714041712bf6
Successfully built labelimg
Installing collected packages: PyQt5-Qt5, PyQt5-sip, pyqt5, lxml, labelimg
```

Figura 36: Instalación de LabelImg  
Imagen elaborada por el autor

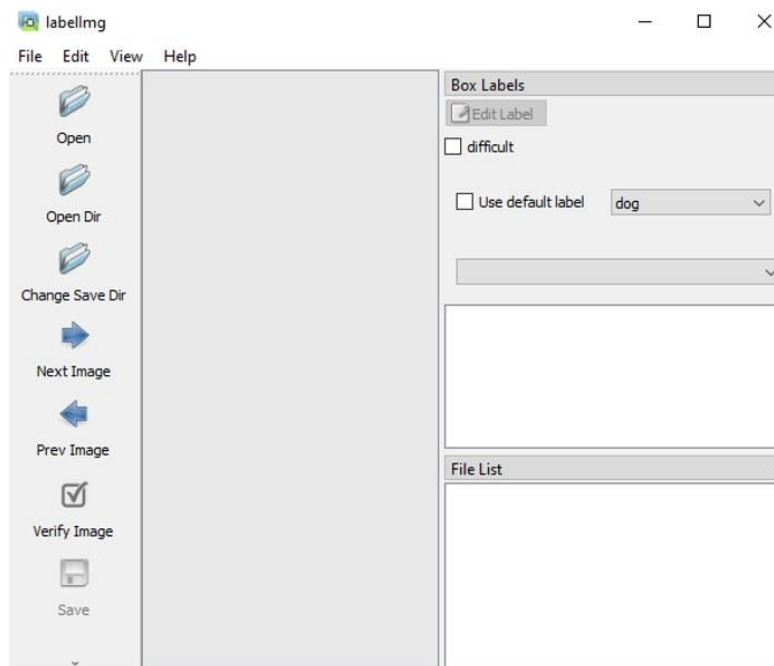


Figura 37: Interfaz de LabelImg  
Imagen elaborada por el autor

Dentro del programa nos dirigimos a **Open Dir**, abrimos la carpeta donde se encuentra nuestro conjunto de datos de imágenes y cargamos las imágenes al programa. Para guardar los datos etiquetados damos en la opción **Change Save Dir** y seleccionamos la misma carpeta donde está el conjunto de datos.

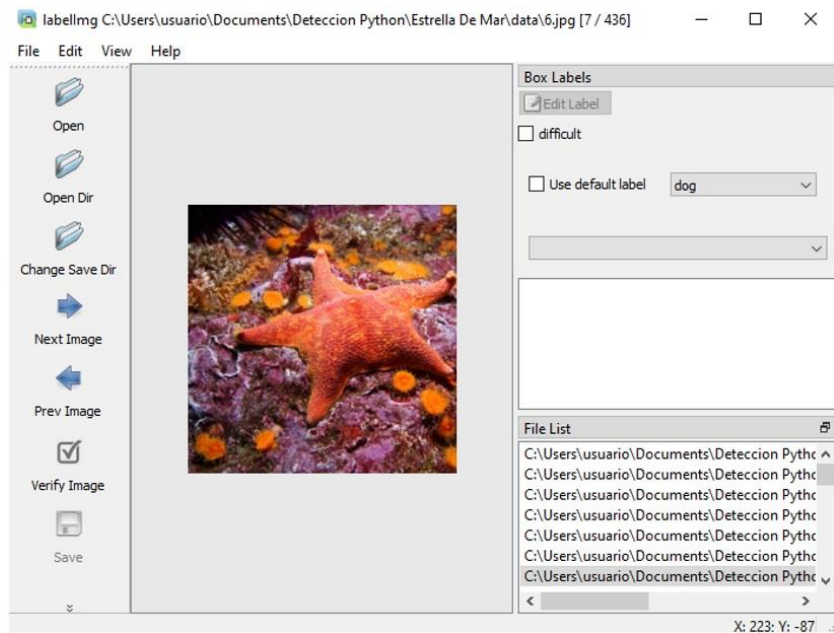


Figura 38: Carga del conjunto de datos a LabelImg

Imagen elaborada por el autor

LabelImg permite guardar las etiquetas en distintos formatos como YOLO, COCO JSON y XML formato PASCAL VOC, como realizaremos el entrenamiento de nuestra red neuronal con YOLO debemos guardar las etiquetas en formato de texto YOLO.

La opción de **Create RectBox**, es la que nos permitirá crear nuestra etiqueta una vez etiquetada la imagen nos aparecerá una venta emergente donde pondremos el nombre de cómo se llamará nuestra clase.

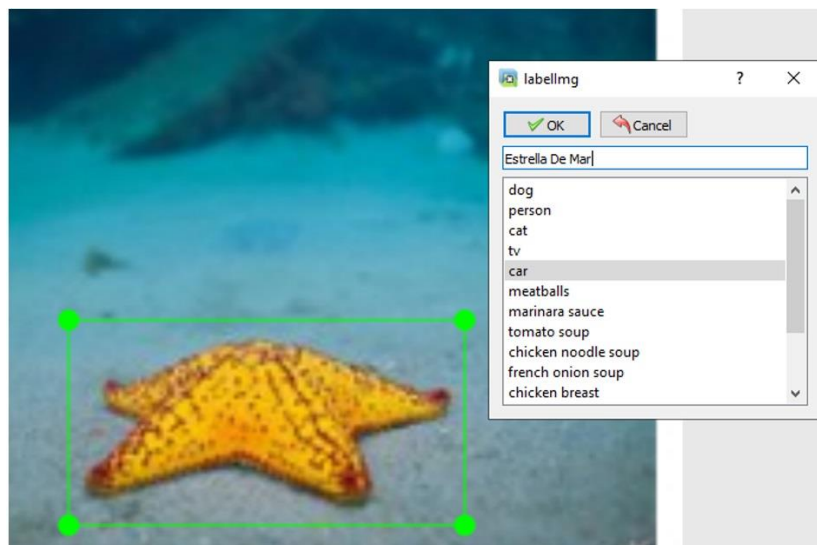


Figura 39: Creación de la etiqueta y la clase

Imagen elaborada por el autor

El proceso de etiquetado tendremos que realizar a todas las imágenes de nuestro conjunto de imágenes, una vez finalizado el etiquetado tendremos listo el conjunto de datos para ser entrenado.

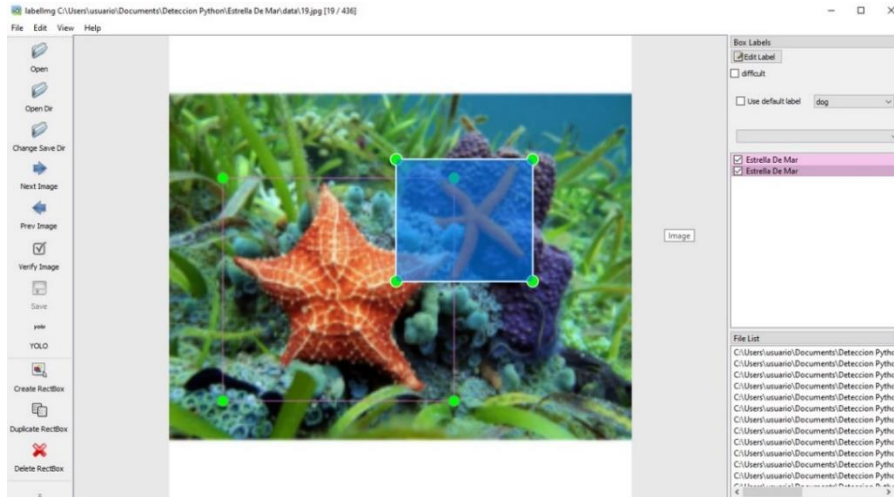


Figura 40: etiquetado de las imágenes  
Imagen elaborada por el autor

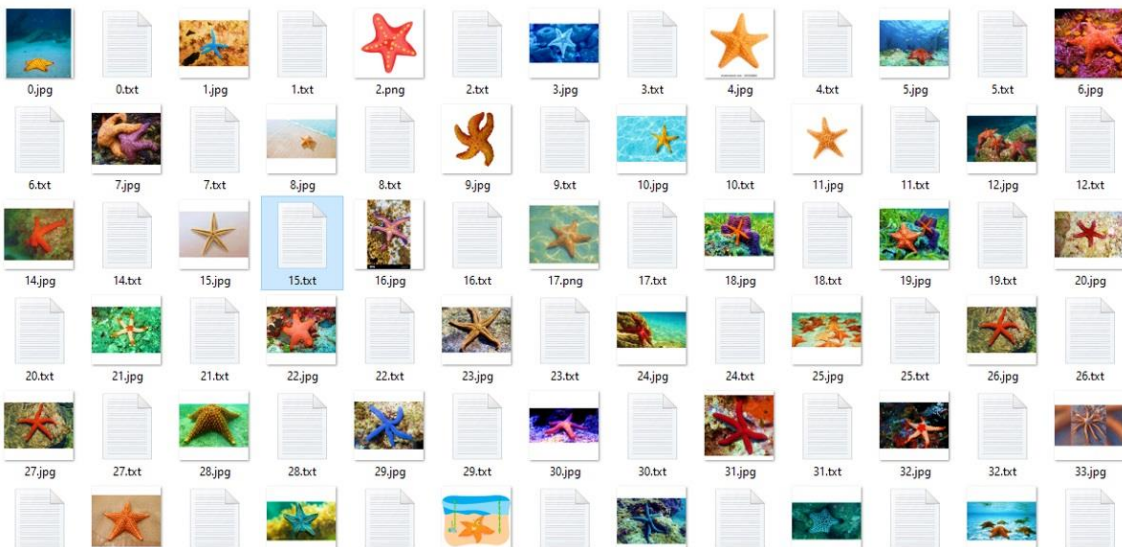


Figura 41: Conjunto de datos con su respectiva etiqueta  
Imagen elaborada por el autor

### 3.1.4. Entrenamiento de la red neuronal personalizada

El entrenamiento de una red neuronal es un proceso robusto que requiere de un dispositivo que tenga grandes capacidades informáticas, por este motivo el entrenamiento no se lo realizara directamente en la Raspberry Pi 4. Google Colab nos ayudara con el entrenamiento desde un ordenador portátil.

#### 3.1.4.1. Preparación de datos

Google Colab es una herramienta de Google Research que permite programar y ejecutar Python desde el navegador. Es de uso gratuito lo cual hace que sea perfecto para desarrollar proyectos de análisis de datos, aprendizaje automático, etc., además, brinda servicio de notebooks de Jupyter permitiendo el uso de acelerador de hardware como CPU, GPU y TPU. Para el entrenamiento de la red neuronal personalizada va ser indispensable el uso de GPU es por eso el uso de Google Colab que nos ofrece el uso de GPU [40]. La Unidad de Procesamiento Grafico (GPU), es muy utilizada con Deep Learning (aprendizaje profundo), debido a la gran capacidad de realizar muchos cálculos matemáticos rápidamente por su arquitectura en paralelo la cual consiste en miles de núcleos más pequeños pero eficientes que se encargan de resolver varias tareas al mismo tiempo, hace a la GPU muy requerida para crear y entrenar modelos de Deep Learning y Machine Learning [41].

Para realizar el entrenamiento necesitaremos de ciertos archivos para entrenar el modelo, estos archivos los podemos encontrar en github <https://github.com/freedomwebtech/yolov4tinypri4>, lo cual procederemos a clonarlos en nuestro ordenador, para aquello abrimos símbolos de sistemas e ingresamos el siguiente comando: *git clone https://github.com/freedomwebtech/yolov4tinypri4.git*

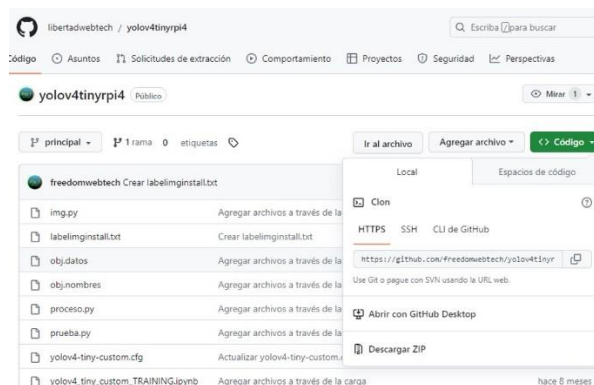


Figura 42: Github repositorio yolov4tinypri4

Imagen elaborada por el autor

```
ca Símbolo del sistema
Microsoft Windows [Versión 10.0.19045.2846]
(c) Microsoft Corporation. Todos los derechos reservados.

C:\Users\usuario>cd C:\Users\usuario\Documents\Deteccion Python

C:\Users\usuario\Documents\Deteccion Python>git clone https://github.com/freedomwebtech/yolov4tinyp4.git
Cloning into 'yolov4tinyp4'...
remote: Enumerating objects: 17, done.
remote: Counting objects: 100% (17/17), done.
remote: Compressing objects: 100% (16/16), done.
remote: Total 17 (delta 4), reused 0 (delta 0), pack-reused 0
Receiving objects: 100% (17/17), 6.44 KiB | 549.00 KiB/s, done.
Resolving deltas: 100% (4/4), done.
```

Figura 43: Clonación del repositorio yolov4tinyp4

Imagen elaborada por el autor

Archivos clonados del repositorio yolov4tinyp4.

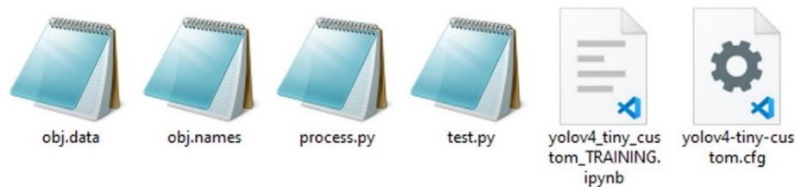


Figura 44: Archivos del repositorio clonado de yolov4tinyp4

Imagen elaborada por el autor

Crearemos una carpeta con el nombre de **yolov4-tiny** en la cual copiaremos los siguientes archivos **obj.data**, **obj.names**, **process.py** y **yolov4-tiny-custom.cfg**, a esta misma carpeta agregaremos nuestro conjunto de datos de imágenes con su respectivo etiquetado en un archivo zip.



Figura 45: archivos que servirán para el entrenamiento

Imagen elaborada por el autor

A los archivos **obj.data**, **obj.names** y **yolov4-tiny-custom.cfg** los editaremos de la siguiente manera. Al archivo **obj.data** en la parte de clases le agregaremos las clases que nuestro entrenamiento va a tener, para este proyecto es una sola clase.



```
obj.data: Bloc de notas
Archivo Edición Formato Ver Ayuda
classes = 1
train = data/train.txt
valid = data/test.txt
names = data/obj.names
backup = /mydrive/yolov4-tiny/training
```

Figura 46: obj.data - número de clases que tendrá nuestra red neuronal

Imagen elaborada por el autor

En **obj.names** agregaremos el nombre de la clase que vamos a entrenar, para este proyecto será “Estrella De Mar”.

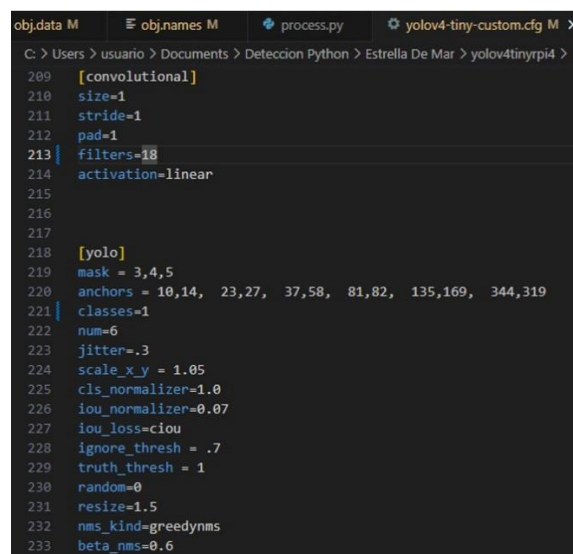


```
obj.names: Bloc de notas
Archivo Edición Formato Ver Ayuda
Estrella De Mar
```

Figura 47: Nombre de la clase que tendrá la red neuronal personalizada

Imagen elaborada por el autor

En el archivo **yolov4-tiny-custom.cfg** editaremos la parte de **classes** que es el número de clases que tendrá nuestro entrenamiento. En la parte de **filters** tendremos que editar dependiendo del número de clases que tengamos **filters=(classes+5) x3** como ya se mencionó anteriormente este proyecto solo detectara una clase entonces quedaría de la siguiente manera **filters= (1+5) x3** que sería igual a 18.



```
obj.data M  obj.names M  process.py  yolov4-tiny-custom.cfg M X
C:\Users\usuario\Documents\Deteccion Python\Estrella De Mar\yolov4tinyrpi4 >
209 [convolutional]
210 size=1
211 stride=1
212 pad=1
213 filters=18
214 activation=linear
215
216
217
218 [yolo]
219 mask = 3,4,5
220 anchors = 10,14, 23,27, 37,58, 81,82, 135,169, 344,319
221 classes=1
222 num=6
223 jitter=.3
224 scale_x_y = 1.05
225 cls_normalizer=1.0
226 iou_normalizer=0.07
227 iou_loss=ciou
228 ignore_thresh = .7
229 truth_thresh = 1
230 random=0
231 resize=1.5
232 nms_kind=greedy_nms
233 beta_nms=0.6
```

Figura 48: Archivo yolov4-tiny-custom.cfg

Imagen elaborada por el autor



Ahora procedemos a subir los archivos de la carpeta **yolov4-tiny** a nuestro repositorio de Google Drive, esto se hace con el fin de que nuestro entrenamiento desde Google Colab pueda acceder al repositorio Drive y extraiga los archivos que se necesitaran en el entrenamiento de red neuronal. Dentro de la carpeta yolov4-tiny de Google Drive crearemos una carpeta con el nombre de *training* donde se guardarán los archivos de nuestro entrenamiento terminado.

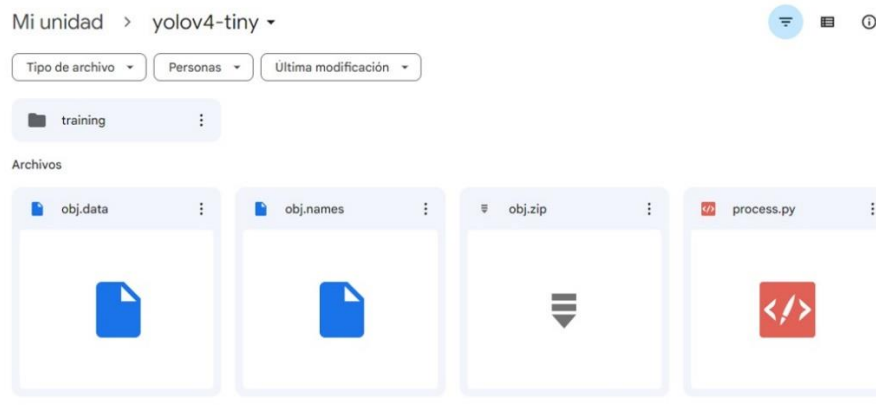


Figura 49: Archivos de entrenamiento cargados a Google Drive.

Imagen elaborada por el autor

### 3.1.4.2. Proceso de entrenamiento

Teniendo listo los archivos que utilizaremos procedemos a acceder a Google Colab y subiremos el archivo **yolov4\_tiny\_custom\_TRAINING.ipynb** el cual contiene los pasos a seguir para entrenar la red neuronal convolucional personalizada.

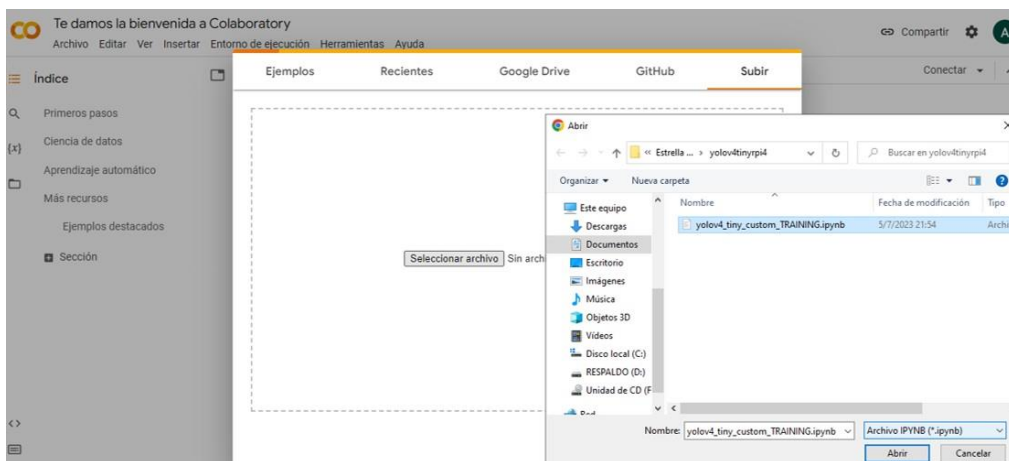


Figura 50: subir archivo a Google Colab

Imagen elaborada por el autor

Ahora debemos configurar el entorno de Google Colab con GPU, nos dirigimos a **Entorno de ejecución | cambiar tipo de entorno de ejecución**, en la parte de tipo de entorno de ejecución pondremos **Python 3** y por último en la parte de Acelerador por hardware seleccionamos **T4 GPU** para que nuestro entorno se ejecute con GPU.

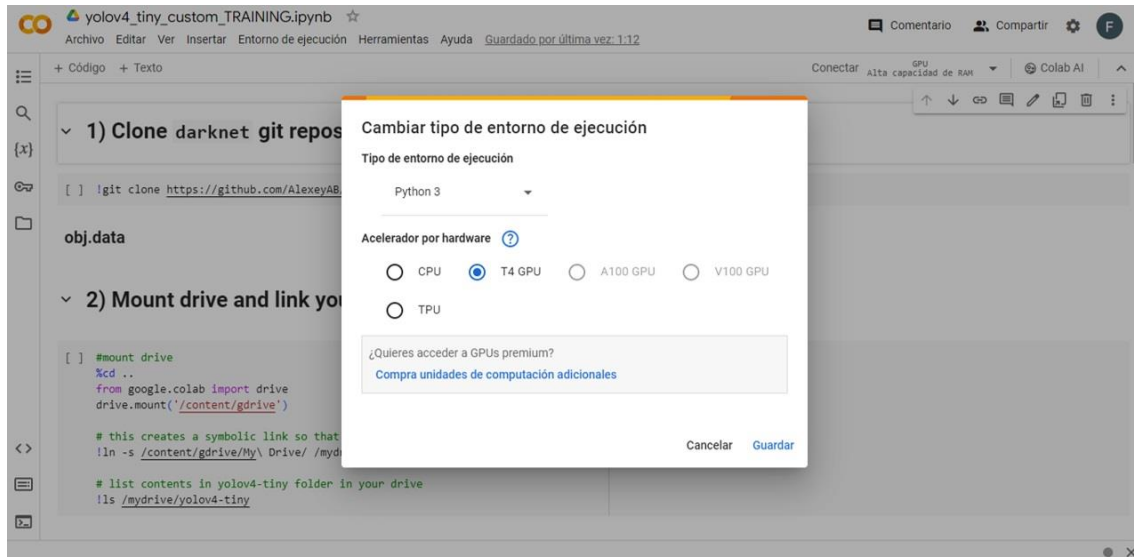


Figura 51: Configuración del entorno de Google Colab

Imagen elaborada por el autor

1. Iniciamos el entrenamiento clonando el repositorio de Darknet el cual servirá como base para nuestro entrenamiento personalizado.

## 1) Clone darknet git repository onto the Colab VM

```
[ ] !git clone https://github.com/AlexeyAB/darknet

Cloning into 'darknet'...
remote: Enumerating objects: 15530, done.
remote: Counting objects: 100% (16/16), done.
remote: Compressing objects: 100% (12/12), done.
remote: Total 15530 (delta 5), reused 13 (delta 4), pack-reused 15514
Receiving objects: 100% (15530/15530), 14.22 MiB | 15.60 MiB/s, done.
Resolving deltas: 100% (10417/10417), done.
```

Figura 52: Clonar repositorio Darknet

Imagen elaborada por el autor

2. Desde Google Colab accedemos al repositorio de Google Drive donde vincularemos la carpeta yolov4-tiny que contiene los archivos que necesitaremos durante el entrenamiento. Cuando ejecutamos esta parte del entrenamiento aparecerá una venta emergente donde tendremos que permitir el acceso a Google Drive.

## 2) Mount drive and link your folder

```
[ ] #mount drive
%cd ..
from google.colab import drive
drive.mount('/content/gdrive')

# this creates a symbolic link so that now the path /content/gdrive/My Drive/ is equal to /mydrive
!ln -s /content/gdrive/My Drive/ /mydrive

# list contents in yolov4-tiny folder in your drive
!ls /mydrive/yolov4-tiny

/
Drive already mounted at /content/gdrive; to attempt to forcibly remount, call drive.mount("/content/gdrive", force_remount=True).
obj.data obj.names obj.zip process.py training yolov4-tiny-custom.cfg
```

Figura 53: Montar Unidad de Google Drive a Google Colab

Imagen elaborada por el autor

3. Ejecutamos esta parte del código para hacer uso de OpenCV, GPU y de cuDNN que es una biblioteca optimizada que contiene implementaciones de GPU para acelerar el rendimiento de la GPU.

## 3) Make changes in the makefile to enable OPENCV and GPU

```
[ ] # change makefile to have GPU and OPENCV enabled
# also set CUDNN, CUDNN_HALF and LIBSO to 1

%cd /content/darknet/
!sed -i 's/OPENCV=0/OPENCV=1/' Makefile
!sed -i 's/GPU=0/GPU=1/' Makefile
!sed -i 's/CUDNN=0/CUDNN=1/' Makefile
!sed -i 's/CUDNN_HALF=0/CUDNN_HALF=1/' Makefile
!sed -i 's/LIBSO=0/LIBSO=1/' Makefile

/content/darknet
```

Figura 54: Habilitar OpenCV y GPU

Imagen elaborada por el autor

4. Como en el primer paso ya clonamos el repositorio de Darknet en nuestro entorno de ejecución ahora podremos construir Darknet que será la columna vertebral de nuestro entrenamiento.

```
4) Run make command to build darknet

[ ] # build darknet
make

In file included from src/yolo_v2_class.cpp:2:
include/yolo_v2_class.hpp: In member function 'void track_kalman::clear_old_states()':
include/yolo_v2_class.hpp:879:50: warning: comparison of integer expressions of different signedness: 'unsigned int' and 'int' [-Wsign-compare]
879 |         if ((result_vec_pred[state_id].x > img_size.width) ||
      |         ~~~~~^~~~~~
include/yolo_v2_class.hpp:880:58: warning: comparison of integer expressions of different signedness: 'unsigned int' and 'int' [-Wsign-compare]
880 |         (result_vec_pred[state_id].y > img_size.height))
      |         ~~~~~^~~~~~
include/yolo_v2_class.hpp: In member function 'track_kalman::tst_t track_kalman::get_state_id(bbox_t, std::vector<bool>&)':
include/yolo_v2_class.hpp:900:38: warning: comparison of integer expressions of different signedness: 'size_t' (aka 'long unsigned int') and 'int' [-Wsign-compare]
900 |         for (size_t i = 0; i < max_objects; ++i)
      |         ~~~~~^~~~~~
include/yolo_v2_class.hpp: In member function 'std::vector<bbox_t> track_kalman::predict()':
include/yolo_v2_class.hpp:909:30: warning: comparison of integer expressions of different signedness: 'size_t' (aka 'long unsigned int') and 'int' [-Wsign-compare]
909 |         for (size_t i = 0; i < max_objects; ++i)
      |         ~~~~~^~~~~~
include/yolo_v2_class.hpp: In member function 'std::vector<bbox_t> track_kalman::correct(std::vector<bbox_t>)':
include/yolo_v2_class.hpp:1025:30: warning: comparison of integer expressions of different signedness: 'size_t' (aka 'long unsigned int') and 'int' [-Wsign-compare]
1025 |         for (size_t i = 0; i < max_objects; ++i)
      |         ~~~~~^~~~~~
src/yolo_v2_class.cpp: In member function 'std::vector<bbox_t> Detector::tracking_id(std::vector<bbox_t>, bool, int, int)':
src/yolo_v2_class.cpp:439:40: warning: comparison of integer expressions of different signedness: 'std::deque<std::vector<bbox_t>>::size_type' (aka 'long unsigned int') and 'int' [-Wsign-compare]
439 |         if (prev_bbox_vec_deque.size() > frames_story) prev_bbox_vec_deque.pop_back();
```

Figura 55: Construir red Darknet

Imagen elaborada por el autor

5. Ahora procedemos a copiar los archivos de la carpeta yolov4-tiny desde Google Drive y los ubica en el directorio de Darknet de Google Colab. Luego se descomprime el archivo zip que contiene el conjunto de datos de imagenes.

### 5) Copy files from your drive to the darknet directory

```
[ ] # Clean the data and cfg folders first except the labels folder in data which is required
%cd data/
!find -maxdepth 1 -type f -exec rm -rf {} \;
%cd ..

%rm -rf cfg/
%mkdir cfg

/content/darknet/data
/content/darknet

[ ] #copy the datasets zip file to the root darknet folder
!cp /mydrive/yolov4-tiny/obj.zip ../

# unzip the datasets and their contents so that they are now in /darknet/data/ folder
!unzip ../obj.zip -d data/
  inflating: data/obj/sketch_27.txt
  inflating: data/obj/sketch_28 - copia.JPEG
  inflating: data/obj/sketch_28 - copia.txt
  inflating: data/obj/sketch_28.JPEG
  inflating: data/obj/sketch_28.txt
```

Figura 56: Copiar archivos de Drive a directorio de Darknet en Google Colab

Imagen elaborada por el autor

6. Se ejecuta el archivo **process.py** y crea los archivos **train.txt** y **test.txt** en el directorio de darknet para poder entrenar nuestra red, train.txt será para entrenar los datos y test.txt como validación de los datos de entrenamiento.

### 6) Run the *process.py* python script to create the *train.txt* & *test.txt* files inside the *data* folder

```
[ ] # run process.py ( this creates the train.txt and test.txt files in our darknet/data folder )
!python process.py

# list the contents of data folder to check if the train.txt and test.txt files have been created
!ls data/

/content/darknet
labels obj obj.data obj.names test.txt train.txt
```

Figura 57: Crear archivos train.txt y test.txt

Imagen elaborada por el autor



carpeta **yolov4-tiny | training** donde encontraremos 6 archivos desde 1000 interacciones a 6000 interacciones.

### 3.2.1. Python

El algoritmo de detección de objetos se desarrollará con el lenguaje de programación Python y con la librería OpenCV, la codificación del código de detección la haremos primero en el ordenador para luego pasar los archivos a la Raspberry Pi 4. Para trabajar con Python desde el ordenador utilizaremos PyCharm, es un entorno de desarrollo integrado que nos permite programar en Python, además ofrece resultado de sintaxis ya que reconoce la sintaxis que utiliza Python lo que permite que sea más legible el código, proporciona completado de código donde brinda sugerencias de código mientras se digita el algoritmo evitando de esta manera la menor cantidad de errores, además permite crear proyectos en entorno virtual facilitando de esta manera el manejo de cada proyecto donde cada entorno virtual maneja los archivos, scripts y bibliotecas instaladas por separado, ya que se podría tener varios entornos con diferentes conjuntos de paquetes sin que exista conflicto entre ellos y sobre todo evita instalar las dependencias de un proyecto en todo el sistema [42].

Ahora debemos crear un proyecto en PyCharm, para realizar esto debemos descargar PyCharm desde su sitio oficial <https://www.jetbrains.com/es-es/pycharm/>, una vez descargado instalamos el programa y lo ejecutamos cuando termine de instalar. En la interfaz de PyCharm nos dirigimos a **File | New Project**, en **Location** buscamos la carpeta donde guardar el proyecto, en **new environment using** colocamos **Virtualenv**, en **Base interpreter** seleccionamos la versión de Python con la que se va a trabajar, para este proyecto trabajaremos con **Python 3.9** y por último damos clic en **Create**.

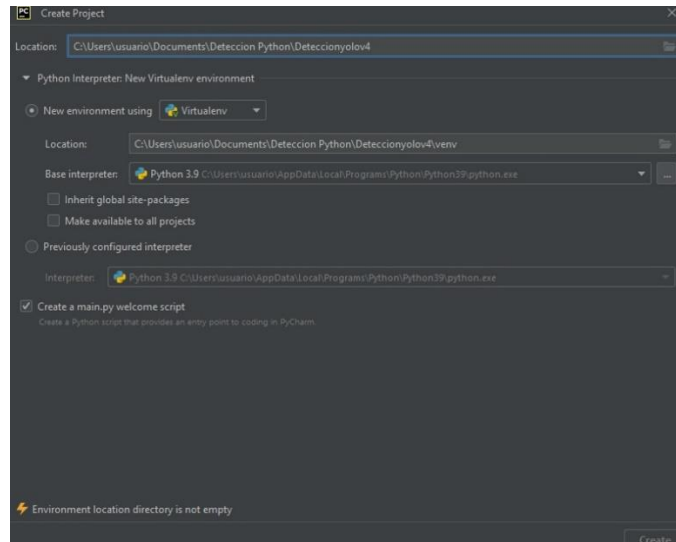


Figura 60: Crear Nuevo proyecto en PyCharm

Imagen elaborada por el autor

Ahora instalaremos las librerías que necesitaremos para ejecutar el algoritmo de detección de objetos. A continuación, se detalla las librerías a instalar desde la terminal de PyCharm.

**Tabla 6: Librerías a instalar para ejecutar la detección de Estrella de Mar.**

| <b>Comandos de librerías a instalar desde la terminal de PyCharm</b> |
|--|
| <code>pip install opencv-contrib-python</code>                       |
| <code>pip install tensorflow</code>                                  |
| <code>pip install matplotlib</code>                                  |
| <code>pip install cvlib</code>                                       |

- OpenCV: Es una librería que dispone de gran cantidad de funciones para el procesamiento y tratamiento de imágenes.
- Cvlib: Es una biblioteca de código abierto usada en visión por computadora, de alto nivel, fácil de utilizar para Python.
- TensorFlow: Es una librería que permite realizar cálculos de vectores de manera eficiente, además es muy utilizada para el desarrollo de aplicaciones de machine learning y deep learning.

- Matplotlib: Es una biblioteca que permite crear gráficos por medio de array y vectores.

Primero debemos importar las librerías que vamos a utilizar: cv2 (librería de OpenCV), os (para poder acceder a los archivos del sistema), cvlib.object.detection import YOLO (ejecuta la inferencia con pesos personalizados de YOLO), cvlib.object.detection import draw\_bbox (permite dibujar el cuadro de la detección).

```
import cv2
import os
from cvlib.object_detection import YOLO
from cvlib.object_detection import draw_bbox
```

En esta parte del código accedemos a los archivos del sistema para guardar las capturas de imágenes la detección del objeto (Estrella de Mar), en caso de que no exista la carpeta "Capturas\_de\_boxes" la crea.

```
dir_name = "Capturas_de_boxes"
if not os.path.exists(dir_name):
    os.mkdir(dir_name)
```

Ahora abrimos el video captura con cap = cv2.VideoCapture(0) y leemos weights o pesos que contiene el modelo personalizado de detección de objetos, después lee el archivo de configuraciones de yolov4-tiny y por último lee el archivo con el nombre de la etiqueta del modelo personalizado (Estrella de Mar).

```
cap = cv2.VideoCapture(0)
weights = "yolov4-tiny-custom_6000.weights"
config = "yolov4-tiny-custom.cfg"
labels = "obj.names"
```

Aquí vamos a escribir cv2.VideoWriter\_fourcc(\*'MP4V') para codificar el video en Python en formato mp4. Con cv2.VideoWriter('filename', fourcc, fps, frameSize) vamos a guardar el video donde: filename (el nombre del video), fourcc (identificador de códec de video), fps (velocidad de fotogramas por segundos que tendrá el video), frameSize (tamaño del video). El count=0 es un contador que servirá más adelante en el código.

```
fourcc = cv2.VideoWriter_fourcc(*'MP4V')
out = cv2.VideoWriter('deteccionEstrella.mp4', fourcc, 10.0, (960, 540))
count = 0
```



Iniciamos un bucle infinito con while True:

```
while True:
```

Realizamos la lectura de la video captura con `ret, img=cap.read()`, donde `ret` (es una variable booleana que nos devuelve un valor de verdadero si la video captura está disponible) y `img` (captura un vector de matriz de imagen de la lectura de video). Cambiamos el tamaño del marco con `cv2.resize`.

```
ret, img = cap.read()
img = cv2.resize(img, (960, 540))
```

llamamos a YOLO gracias a `cvlib` que nos permite ejecutar inferencia con pesos personalizados, dentro de YOLO colocamos `weights` (contiene el modelo personalizado de detección de objeto), `config` (contiene la configuración de `yolov4-tiny`), `labels` (contiene el nombre de la etiqueta del modelo personalizado).

```
yolo = YOLO(weights, config, labels)
```

realizamos la detección con `yolo.detect_objects(img)` y guardamos en `bbox` (el cuadro donde se encuentra el objeto), `label` (guardamos la etiqueta del objeto) y en `conf` (guarda la confianza del objeto detectado). Con `print(bbox, label, conf)` presentamos las coordenadas del cuadro donde se encuentra el objeto detectado, se presenta la etiqueta y la confianza que tiene el objeto detectado.

```
bbox, label, conf = yolo.detect_objects(img)
print(bbox, label, conf)
```

con `yolo.draw.bbox (img, bbox, label, conf, write_conf=True)`, vamos a dibujar una caja delimitadora alrededor de los objetos detectados en la video captura, donde `img` (variable donde esta guardada la video captura), `bbox` (contiene las coordenadas del objeto), `label` (contiene el nombre de la etiqueta del objeto), `conf` (contiene la confianza del objeto detectado) y `write_conf=True` (escribe la confianza en la caja delimitadora).

```
yolo.draw_bbox(img, bbox, label, conf, write_conf=True)
```

colocamos la siguiente condición `if conf >= [0.8286921381950378]`, para guardar las imagenes detectados con objetos que superen la confianza en 0.82, esto se hace para que no se guarden falsos positivos en la detección de objeto (Estrella de Mar). Con `cv2.imwrite(dir_name + '/objeto_{}.jpg'.format(count), img)` guardamos las imagenes en la carpeta que se creó al inicio del código y colocamos un contador `count = count + 1` para que guarden sin límite las imágenes. El guardado de las imágenes se realiza con la finalidad de tener evidencia de la detección del objeto.

```
if conf >= [0.8286921381950378]:
```

```
    cv2.imwrite(dir_name + '/objeto_{}.jpg'.format(count), img)
    count = count + 1
```

con `cv2.imshow("Deteccion", img)` presentamos la video captura en tiempo real con la detección de objeto, `out.write(img)` guardamos el video de la detección de objeto para evidencia de la detección. Ahora vamos a leer el teclado y cerrar la venta de video captura en tiempo real con `if cv2.waitKey(1) & 0xFF == 27` y salimos el bucle `while True`.

```
cv2.imshow("Deteccion",img)
out.write(img)
if cv2.waitKey(1) & 0xFF == 27:
    break
```

por ultimo con `cap.release()` borramos la video captura y con `cv2.destroyAllWindows()` cerramos todas las ventanas que se abrieron durante la ejecución del scripts de la detección de objeto.

```
cap.release()
cv2.destroyAllWindows()
```

### 3.2.2. Adaptación del modelo entrenado a la Raspberry Pi 4

La Raspberry Pi 4 será la encargada de ejecutar el sistema de visión artificial del robot submarino para la detección de la especie marina Estrella de Mar. Para el funcionamiento de la Raspberry Pi 4 se realiza la instalación del sistema operativo Raspberry Pi OS 64-bit Debian Bullseye 2023-05-03, en la sección de anexos se explica de manera detalla la instalación del sistema operativo para la Raspberry Pi 4.

Para poder ejecutar el sistema de visión artificial es necesario instalar algunas librerías en la terminal de la Raspberry, las siguientes librerías son:

```
pip install opencv-contrib-python
```

```
pip install tensorflow
```

```
pip install cvlib
```

```
pip install matplotlib
```

Ahora se habilitará la cámara USB del robot submarino que se encuentra conectada a la Raspberry, para realizar aquello necesitamos instalar algunas librerías.

```
sudo apt-get install -y cmake libjpeg9-dev
```

Mjpg-streamer nos ayudara copiando fotogramas de una o varios complementos de entrada a múltiples complementos de salida permitiendo transmitir archivos JPEG por medio de una red establecida mediante IP. Con los siguientes comandos se descarga y descomprime el archivo para instalar mjpg-streamer.

```
wget https://github.com/jacksonliam/mjpg-streamer/archive/master.tar.gz
```

```
tar -xvf master.tar.gz
```

Teniendo los archivos listos para instalar accedemos a la carpeta donde se encuentran los archivos y con el comando make instalamos mjpg-streamer.

```
cd mjpg-streamer-master/mjpg-streamer-experimental
```

```
make
```

```
sudo make install
```

una vez instalado mjpg-streamer ejecutamos el comando que habilita la cámara que servirá para la detección de la especie marina “Estrella de mar”.

```
./mjpg_streamer -i "input_uvc.so -d /dev/video0 -r 640x480 -f 30" -o "output_http.so -w  
./www"
```

Para comprobar que la transmisión de video este funcionado correctamente ingresamos el siguiente comando en el navegador de su preferencia.

*http://direccion\_ip\_de\_la\_raspberry:8080*

Como el robot submarino se conecta a la estación de control por medio de una conexión cliente-host, el robot utilizara la siguiente IP para poder acceder al video

*http://192.168.2.2:8080/?action=stream*

Esta dirección debe ir en el código para la detección de la especie marina Estrella de mar en la parte de *cap = cv2.VideoCapture(http://192.168.2.2:8080/?action=stream)*

Teniendo lista la Raspberry para ejecutar el sistema de visión artificial, procedemos a transferir los archivos desde la PC a la Raspberry para poder ejecutar el código de detección. El programa WinSCP nos ayudara con este proceso, teniendo instalado el programa en la PC lo ejecutamos y tendremos que ingresar lo siguiente: En la parte de protocolo seleccionamos SCP; en Nombre o IP del servidor ingresamos la IP de la Raspberry; en el número de puerto dejamos 22; el usuario y contraseña ingresamos la que colocamos cuando habilitamos el SSH y damos en conectar.

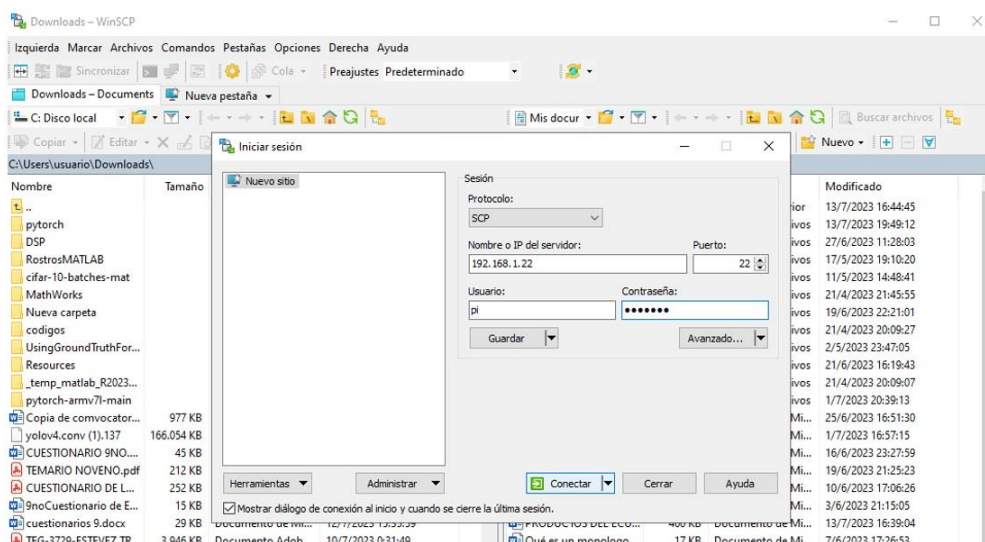


Figura 61: Comunicación entre WinSCP y la Raspberry

Imagen elaborada por el autor

Si ingresamos todo correctamente aparecerá la siguiente interfaz. Para transferir los archivos simplemente se arrastra la carpeta o archivo desde la venta izquierda de la PC a la venta derecha de la Raspberry.

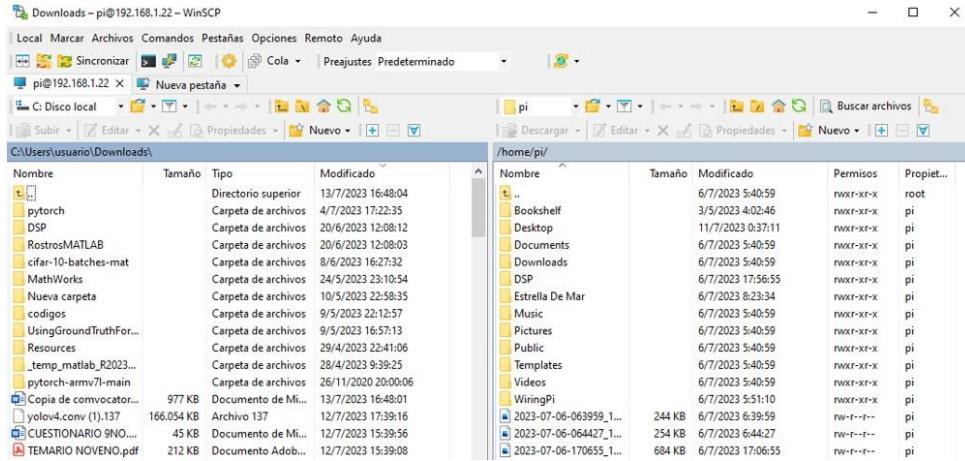


Figura 62: Trasferir archivos entre la PC y Raspberry con WinSCP  
Imagen elaborada por el autor

## Capítulo IV

### 4. Diseño y Configuración del Sistema de Navegación Autónoma

En el presente capítulo se aborda todo lo relacionado con el sistema de navegación autónoma, donde se detallará el proceso a seguir para diseñar el sistema de control del robot submarino.

#### 4.1. Mission Planner vs QGroundControl

Actualmente los softwares Mission Planner y QGroundControl permiten conectarse a cualquier tipo de vehículo no tripulado, además proporcionan los dos firmwares principales de código abierto que son el APM creado por el proyecto ArduPilot y PX4 creado por el proyecto PX4-Autopilot. Hoy en día tanto Mission Planner y QGroundControl permiten trabajar con los dos tipos de firmwares sin que repercuta en el funcionamiento del vehículo.

Tanto el software Mission Planner y QGroundControl permiten controlar, planificar misiones y configurar parámetros con los dos firmwares principales para el control de vehículos no tripulados, sin que se pierda la configuración que se realizó en un software cuando se conecte el robot al otro software. Por esta razón se utilizará ambos softwares para planificar misiones y configurar los parámetros del robot submarino y que este pueda realizar la navegación autónoma de manera eficiente. El mejor software para trabajar con el robot submarino es QGroundControl ya que proporciona alertas de seguridad durante las misiones y al tener acceso al modo manual permite cancelar y actuar rápidamente en caso de emergencia.

**Tabla 7: Características del Software Mission Planner y QGroundControl**

| <b>Características</b>           | <b>Mission Planner</b>   | <b>QGroundControl</b>  |
|----------------------------------|--|--|
| <b>Licencia</b>                  | Licencia Open Source GPLv3.  | Licencia Open Source GPLv3.                                    |
| <b>Plataformas</b>               | Windows y soporte limitado para Mac OS, Linux y Android.             | Windows, Mac OS, Linux, Android, iOS.                          |
| <b>Compatibilidad</b>            | Compatible con ArduPilot y PX4.                                      | Compatible MAVLink incluyendo ArduPilot y PX4.                 |
| <b>Interfaz de usuario</b>       | Posee más opciones y configuraciones avanzadas, dificultando su uso. | Posee una interfaz intuitiva de fácil manejo.                  |
| <b>Planificación de misiones</b> | Planificación de misiones completa y detallada.                      | Planificación de misiones fácil y rápida de realizar           |
| <b>Uso en campo</b>              | No es muy portátil   | Es una herramienta portátil, fácil manejo en tablets y móviles |
| <b>Funciones adicionales</b>     | Proporciona simulación (multirotor, helicopter, plane y rover)       | No proporciona simulación                                      |

**Tabla 8: Ventajas y Desventajas del Software Mission Planner y QGroundControl**

|                    | <b>Mission Planner</b>  | <b>QGroundControl</b>  |
|--------------------|---|--|
| <b>Ventajas</b>    | <ul style="list-style-type: none"> <li>• Ofrece gran variedad de configuraciones avanzadas, permitiendo un control detallado sobre el comportamiento del vehículo.</li> <li>• Proporciona configuraciones detalladas y análisis de datos, lo que facilita una evaluación exhaustiva de las misiones realizadas.</li> <li>• Brinda actualizaciones de firmware del piloto automático, además permite trabajar con versiones anteriores y en modo desarrollador.</li> <li>• Cuenta con foros de ayuda dentro de la comunidad de ArduPilot.</li> </ul> | <ul style="list-style-type: none"> <li>• Brinda una interfaz moderna, intuitiva y fácil de usar facilitando el uso para principiantes.</li> <li>• Compatible con múltiples plataformas como Windows, Mac OS, Linux, Android, iOS lo que facilita su uso en campo ya que es compatible con dispositivos móviles.</li> <li>• Fácil planificación de misiones autónomas.</li> <li>• Proporciona actualización de firmware, permitiendo trabajar en modo desarrollador y versiones antiguas de firmware.</li> <li>• Cuenta con foros de ayuda y soporte técnico dentro de la comunidad de BlueRobotics.</li> </ul> |
| <b>Desventajas</b> | <ul style="list-style-type: none"> <li>• Dificulta de uso, la interfaz puede ser muy compleja dificultando el uso para principiantes.</li> <li>• No es muy conveniente para uso de campo su interfaz en herramienta móvil es muy compleja y con errores.</li> <li>• Aunque cuenta con simulación no está disponible para Submarino.</li> </ul>  | <ul style="list-style-type: none"> <li>• No proporciona configuraciones avanzadas en comparación con Mission Planner.</li> <li>• No proporciona simulación</li> </ul>  |



## 4.2. Implementación del Sistema de Control

### 4.2.1. Arquitectura del sistema de navegación

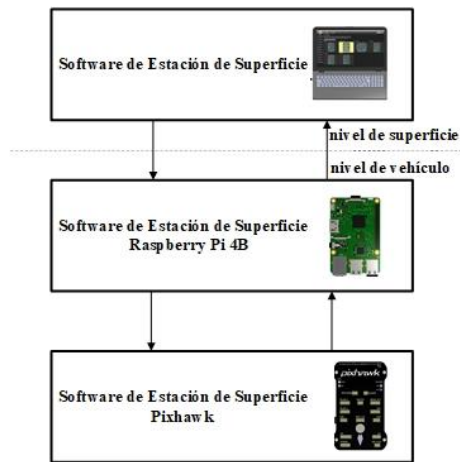


Figura 63: Arquitectura general del Robot Submarino

Imagen elaborada por el autor

La arquitectura general del sistema de navegación está compuesta por tres niveles. Nivel inferior donde se encuentra el controlador Pixhawk en el cual se ejecuta el firmware ArduSub que se encarga de controlar los motores, sensores y servomotores del robot submarino. En el nivel intermedia encontramos a la Raspberry Pi que se encarga de ejecutar el sistema operativo del robot submarino y actúa como middleware entre la capa inferior del controlador Pixhawk y la estación de control. En el nivel superior encontramos la estación de control, donde se ejecuta el software QGroundControl o Mission Planner.

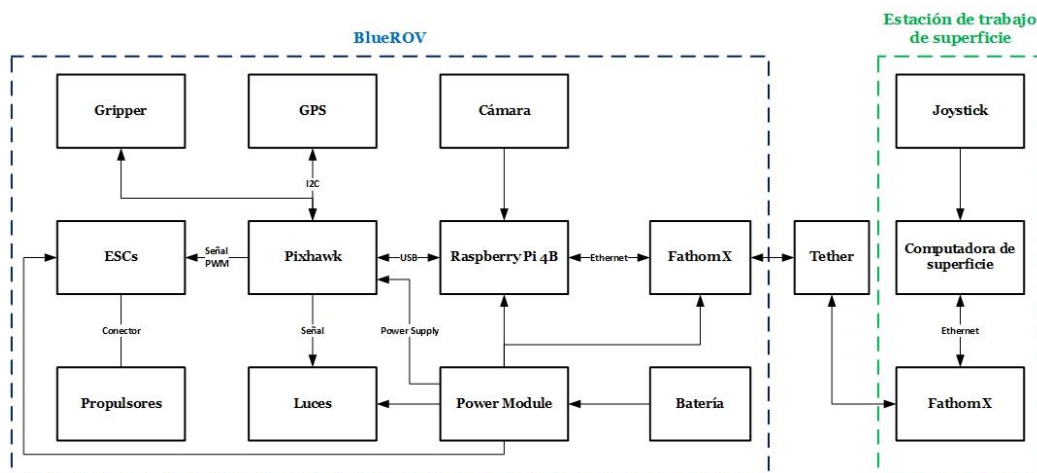


Figura 64: Arquitectura de la electrónica del sistema de navegación del robot submarino

Imagen elaborada por el autor

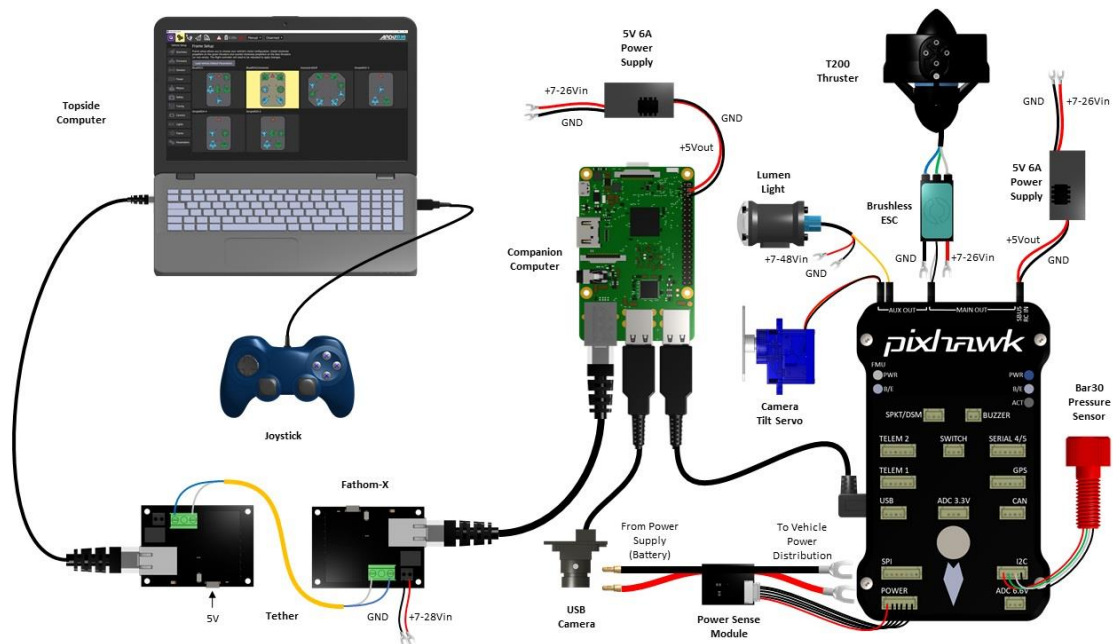


Figura 65: Diagrama de conexiones del robot submarino [18].

La integración de los componentes que conforman al robot submarino juega un papel crucial para el correcto funcionamiento del sistema de control. El robot submarino utiliza el controlador Pixhawk donde se ejecuta el firmware ArduSub el cual permite gestionar la estabilidad y el control del robot, esto es posible ya que controla los motores y además procesa los datos de los sensores y la información enviada desde la estación de control.

La Raspberry Pi se encarga de ejecutar el sistema operativo BlueOS, el cual se encarga de ejecutar dos procesos fundamentales para el funcionamiento del robot: el primero se encarga de establecer la comunicación entre la estación de control y el nivel inferior donde se encuentra el controlador Pixhawk; el segundo proceso se encarga de ejecutar la transmisión de video desde la cámara del robot a la estación de control que recibe fotogramas comprimidos por medio de UDP, la comunicación de estos procesos es a través del cable Tether y la Fathom-X lo que permite el envío de información entre ambos niveles.

La estación de control en superficie se ejecuta el software QGroundControl o Mission en el cual se reciben datos de video, se puede monitorear el comportamiento del robot en tiempo real y alertas de emergencias enviadas desde el robot submarino. Además, desde la estación de control se planifican las misiones autónomas que vaya a realizar el robot y que este pueda entrar en modo estable para realizar el proceso de detección de la Estrella de Mar.

El GPS cumple un rol importante dentro del sistema de navegación del robot submarino, el cual va a permitir que el robot pueda cambiar a modos de navegación que sin este no son posibles ejecutar. El GPS es fundamental para la ejecución de misiones autónomas y que este tenga mejor estimación de posición geodésica en modo estable. El GPS deberá ir conectado en el puerto GPS y el I2C de la Pixhawk. A continuación, se muestra la conexión del GPS.

**Tabla 9 Conexión de GPS a la placa Pixhawk**

| <b>Puerto GPS</b> |              |                    |
|-------------------|--------------|--------------------|
| <b>Pin</b>        | <b>Color</b> | <b>Descripción</b> |
| 1                 | Rojo         | 5 V                |
| 2                 | Amarillo     | GPS RX             |
| 3                 | Verde        | GPS TX             |
| 4                 | No conecta   | No conecta         |
| 5                 | No conecta   | No conecta         |
| 6                 | Negro        | GND                |
| <b>Puerto I2C</b> |              |                    |
| 1                 | No conecta   | No conecta         |
| 2                 | Rojo         | I2C SCL            |
| 3                 | Azul         | I2C SDA            |
| 4                 | No conecta   | No conecta         |

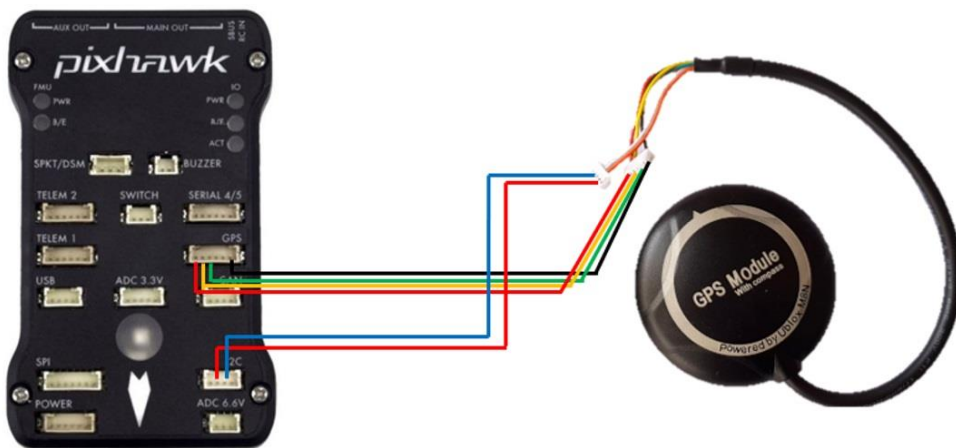


Figura 66: Conexión de GPS a la Pixhawk

Imagen elaborada por el autor

En la figura 67 se puede apreciar cómo se integra la comunicación entre los 3 niveles de la arquitectura general antes mencionada. Para que esta comunicación sea posible primero se deberá establecer ip estática a la Raspberry (192.168.2.2) y a la computadora de superficie (192.168.2.1), en el apartado de anexos encontrara de manera detallada como establecer la ip estática a la Raspberry y la computadora de superficie. La comunicación del robot submarino se realiza utilizando el protocolo MAVLink, al momento de enviar un mensaje desde la estación de control, los mensajes MAVLink son encapsulados en datagramas y se envían a la Raspberry Pi por medio del cable Tether que conecta dos tarjetas Fathom-X una se encuentra en el robot submarino y la otra en la superficie que conecta a la estación de control. La Raspberry se encarga de descryptar los mensajes de los paquetes UDP por medio de MAVProxy y reenvía los mensajes hacia la Pixhawk que se encuentra conectada a través de un puerto USB. La Pixhawk también envía mensajes MAVLink al MAVProxy como respuestas a comandos de alto nivel a la estación de control o estados del robot que pueden ser información del IMU, nivel de batería o la posición del robot la cual esta filtrada por el Filtro de Kalman Extendido (EKF), este combina los datos del sensor de profundidad, brújula, el IMU (Unidad de medición inercial) y las actualizaciones de posición brindada por el GPS, las cuales pueden prevenir tanto de mensajes MAVLink como de un GPS directamente conectado al controlador Pixhawk.

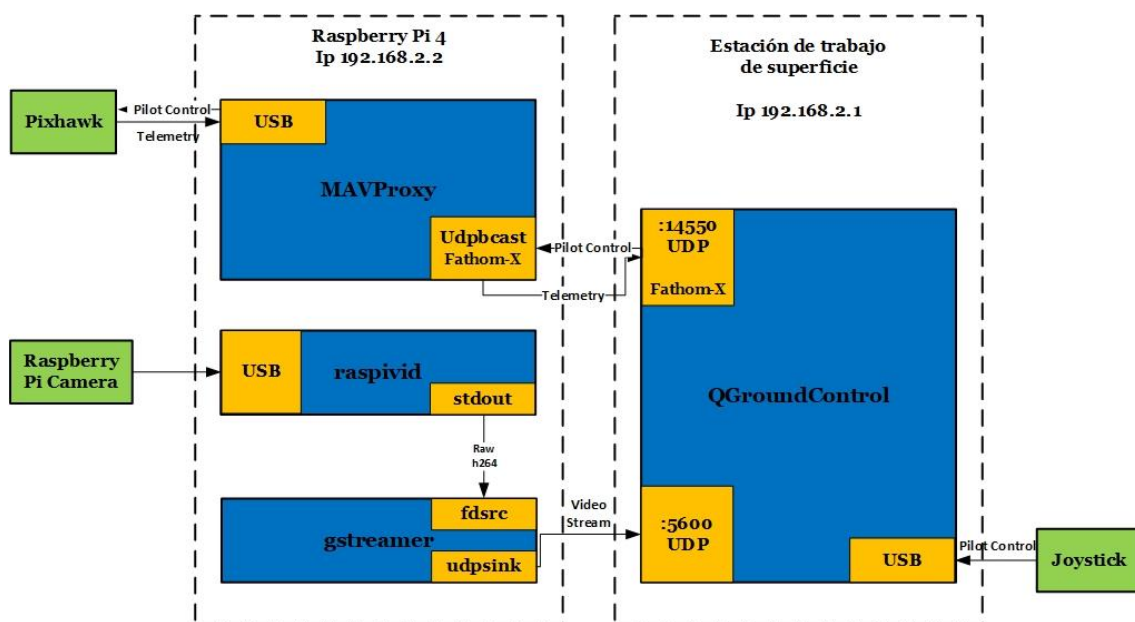


Figura 67: Diagrama de funcionamiento de los módulos de comunicación de software

Imagen elaborada por el autor

Teniendo claro cómo se integran los componentes electrónicos del sistema de navegación y control del robot submarino hablaremos porque se decidió trabajar con el controlador Pixhawk PX4 y la Raspberry Pi 4. Si bien es claro en la Universidad Estatal Península de Santa Elena ya se ha realizado proyectos de robot submarino, pero ninguno de ellos utiliza como controlador a la Pixhawk. La Pixhawk está diseñada (para proporcionar un sistema de control) con el objetivo de facilitar control de los vehículos autónomas ya que incluye sensores integrados como IMU, acelerómetro y giroscopio para medir los movimientos y orientación del vehículo, además emplea el firmware ArduSub el cual se carga en la memoria interna del Pixhawk.

Este firmware ArduSub contiene la lógica para el control y navegación del robot submarino, además permite la transmisión y recopilación de datos en tiempo real, inicialmente se trabajaba con el firmware 3.5.0 pero este no permitía cambiar a modo Auto y Guided es por esta razón que se cambió a la versión 4.0.1 que incluye varias correcciones y mejoras que versiones anteriores además admite cambio a modo Auto y Guided cuando se coloca el GPS, lo cual ayuda al robot a tener mejor control y estabilidad.

También se reemplazará el sistema operativo de la Raspberry Pi, anteriormente se utilizaba ArduSub Companion ahora se pasará al BlueOS que es la versión actual de sistema operativo para vehículos submarinos, además cuenta con una interfaz web donde se puede monitorear y configurar el robot submarino, también se cambiará la Raspberry Pi 3B por la Raspberry Pi 4B para poder ejecutar el proceso de detección de la Estrella de Mar, ya que con la Pi 3 no permitía la compilación del sistema operativo BlueOS y el sistema de reconocimiento de la especie marina.



Figura 68: Cambio de Raspberry y Pixhawk

Imagen elaborada por el autor

Como se realizó el cambio Raspberry y Pixhawk en la figura 68 se presenta el diagrama de las conexiones entre la Pixhawk y los motores del submarino. En la siguiente tabla se especifica en que pines deben ir conectados los motores y servomotores.

**Tabla 10. Conexión de motores a la placa Pixhawk**

| Pines Principales |                                   |
|-------------------|-----------------------------------|
| Canal de salida   | BlueROV                           |
| 1                 | Propulsor 1                       |
| 2                 | Propulsor 2                       |
| 3                 | Propulsor 3                       |
| 4                 | Propulsor 4                       |
| 5                 | Propulsor 5                       |
| 6                 | Propulsor 6                       |
| 7                 | Luces                             |
| 8                 | Servo de inclinación de la cámara |
| SBUS (SB)         | Entrada de 5 V                    |
| RCIN (RC)         | No utiliza                        |
| Pines Auxiliares  |                                   |
| 3                 | Pinza                             |
| 6                 | Sensor                            |

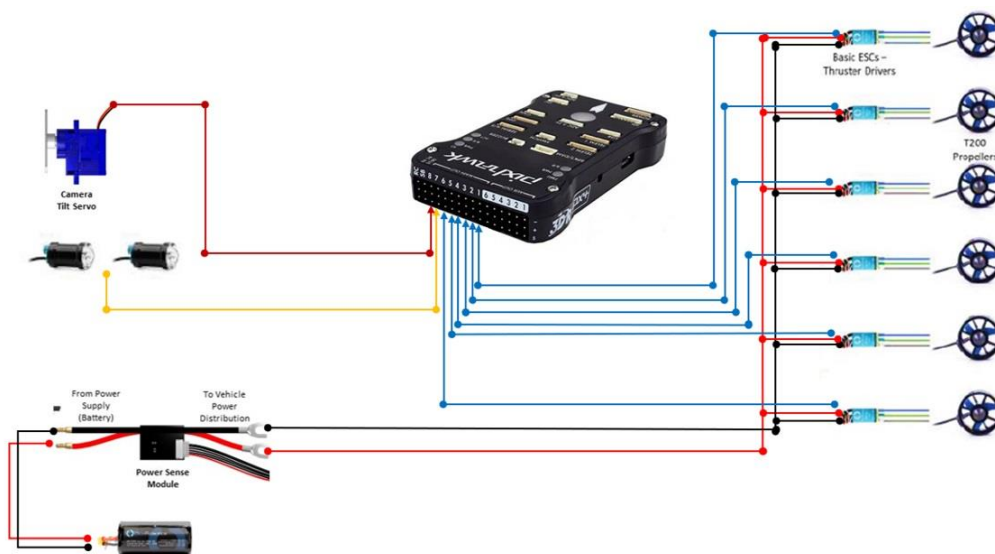


Figura 69: Conexión de motores a la Pixhawk

Imagen elaborada por el autor

Otra conexión a tener en cuenta es entre la Raspberry con la Pixhawk, se comunican por medio un cable USB que permite el envío de información entre el sistema operativo con la Pixhawk.

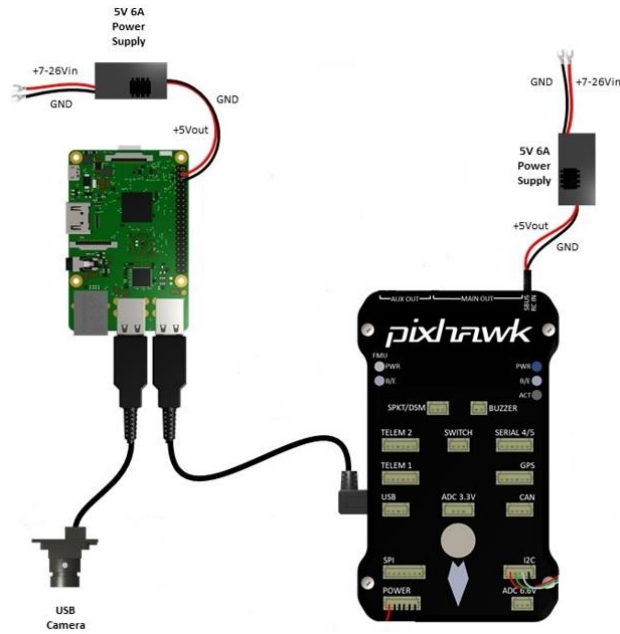


Figura 70: Conexión entre la Raspberry y la Pixhawk [18].

### 4.3. Configuración y Personalización

#### 4.3.1. Personalización del software BlueOS

La Raspberry Pi 4 ya cuenta con un sistema operativo Raspberry Pi OS 64-bit Debian Bullseye en el cual se ejecuta el proceso de detección de la Estrella de Mar. Para que el robot submarino sea capaz de entablar comunicación con la estación de control es necesario sistema operativo BlueOS siendo una parte fundamenta en el sistema de navegación y control de robot, ya que se encarga de integrar el controlador Pixhawk, la Raspberry Pi y diversos sensores, además se asegura que todos los sistemas trabajen de manera conjunta y por medio de su interfaz web facilita la configuración y monitoreo del robot submarino. ¿Cómo instalar el sistema operativo BlueOS si la Raspberry Pi ya cuenta con un sistema operativo? Para instalar el sistema operativo es necesario hacerlo en modo desarrollador, nos dirigimos al repositorio github de BlueRobotics <https://github.com/bluerobotics/BlueOS/tree/master/install> el cual contiene todos los archivos necesarios para la instalación de BlueOS-Docker. Para iniciar con la instalación de BlueOS en la terminal de la Raspberry ejecutamos el siguiente comando, una vez terminado de instalar se debe reiniciar la Raspberry para que se efectúen los cambios:

```
sudo su -c 'curl -fsSL https://raw.githubusercontent.com/bluerobotics/blueos-docker/master/install/install.sh | bash'
```

La configuración de parámetros tales como calibración del acelerómetro, compass dentro del software QGroundControl o Mission Planner, cumplen un papel importante para el control y navegación del robot ya que si estos parámetros no se encuentran en una correcta calibración el robot puede presentar fallas al momento de ejecutar alguna misión o simplemente no se podrá ejecutar ninguna misión por los parámetros de seguridad que no permiten encender los motores sin que el robot se encuentre correctamente configurado.

Antes de iniciar cualquier calibración se debe cargar el firmware ArduSub versión estable 4.0.1 al controlador Pixhawk, la instalación del firmware ArduSub es de gran importancia ya que este contiene la lógica necesaria para el control y navegación del robot submarino. Esta versión la podemos descargar en el siguiente link: <https://www.ardusub.com/resources/downloads.html>. Para instalar el firmware con Mission Planner conecte la Pixhawk a la computadora con un cable USB, luego vaya a **Setup | Install Firmware** aparecerá una interfaz donde podrá elegir qué tipo de firmware instalar en su versión actualizada, como trabajaremos con la versión 4.0.1 haga clic en la



parte de **load custom firmware** y seleccione el firmware descargado anteriormente y empezara la instalación del firmware, para saber de manera detallada como instalar el firmware en anexos encontrara una sección sobre la instalación del firmware ArduSub.

### 4.3.2. Calibración de los sensores de la Pixhawk

La calibración del Pixhawk es un parámetro a configurar muy necesario para corregir variaciones en la orientación del robot submarino. Para realizar la calibración tanto del acelerómetro y del Compass utilizaremos Mission Planner, además el robot ya debe tener conectado el GPS a la Pixhawk. Antes de calibrar los sensores de la Pixhawk debemos configurar la orientación que tendrá la Pixhawk que debe ser Roll90, para cambiar la orientación nos dirigimos a **Configuración | Full Parameter List**, en la lista de parámetros buscamos **AHRS\_ORIENTATION** y colocamos el parámetro **16: Roll90**, para guardar estos cambios presionamos **Write Params**.

#### 4.3.2.1. GPS

El GPS será el encargado de establecer una posición de referencia al robot submarino además ayuda a mejorar la funcionalidad y precisión de los sensores integrados con los que cuenta el Pixhawk. El GPS permite al robot submarino cambiar de modo autónomo y guiado, lo cual es importante para la planificación y ejecución de la navegación autónoma que realizara el robot submarino, que se hará por medio de puntos de referencia (Waypoint) ya que sin el GPS el robot submarino solo podría funcionar en modo manual.

Para que el robot submarino pueda recibir información del GPS se debe incluir en el EKF (filtro de Kalman Extendido). Incluir el GPS en los parámetros EKF (Filtro de Kalman Extendido) el cual combina los datos del GPS con los sensores integrados como el acelerómetro, giroscopio y magnetómetro mejorando la precisión y estabilidad del sistema de navegación y control del robot submarino, por lo que se debe configurar los siguientes parámetros:

- En **Mission Planner | CONFIG** seleccionamos **Full Parameter List** y modificamos el parámetro **SERIAL3\_PROTOCOL** ubicamos el valor **2** para configurar como puerto de entrada de comunicación GPS por medio de MAVLink.
- Para que se incluya el GPS en el EKF se debe modificar el parámetro **GPS\_TYPE** donde ubicaremos el valor de **14**.

#### 4.3.2.2. Calibración del acelerómetro

Para calibrar el acelerómetro en Mission Planner nos dirigimos a **SETUP | Mandatory Hardware**, luego seleccionamos **Accel Calibration** y seguimos las indicaciones que proporciona el programa. En la sección de anexos encontrara de manera detallada la calibración del acelerómetro.

Calibrar el acelerómetro es crucial para el funcionamiento del robot, proporciona datos de la aceleración en los ejes X, Y y Z. Los datos de aceleración son fundamentales para la navegación y el control del robot submarinos. A continuación, se detalla los tipos de datos que proporciona el acelerómetro:

- **Aceleración lineal:**
  - Eje X: En el eje X mide la aceleración hacia adelante y atrás.
  - Eje Y: En el eje Y mide la aceleración hacia los lados, izquierda y derecha.
  - Eje Z: En el eje Z mide la aceleración hacia arriba y hacia abajo.
- **Orientación y posición:** Los datos de aceleración en los tres ejes X, Y y Z se combinan con los datos del giroscopio, magnetómetro y GPS para determinar la posición y orientación del robot submarino en el espacio.
- **Detección de movimientos:** El acelerómetro detecta cambios de velocidad del robot submarino, estos datos son utilizados para mantener estable al robot durante el movimiento y ajustar la velocidad de los propulsores según sea necesario.
- **Precisión de datos:** realizar la calibración del acelerómetro garantiza que las lecturas que toma sean precisas, lo cual es esencial para el sistema de control y navegación del robot, ya que cualquier error de medición de la aceleración puede ocasionar desviaciones en la orientación y trayectoria del robot.
- **Estabilidad del sistema:** Una correcta calibración del acelerómetro ayuda a la estabilidad del sistema de control del robot submarino, los datos proporcionados son utilizados para ajustar los propulsores con el objetivo de estabilizar al robot submarino durante la misio autónoma.
- **Integración con otros sensores:** al tener como controlador a la Pixhawk que posee varios sensores integrados los datos del acelerómetro se combinan con sensores como el giroscopio y magnetómetros, además de la incorporación del

GPS. La calibración del acelerómetro asegura que todos los sensores internos y externos trabajen juntos de manera coherente

#### 4.3.2.3. Calibración del Compass

Para realizar la calibración del compass (brújula) vaya a **Configuración | Mandatory Hardware**, luego seleccione **Compass**. Como se van a calibrar la brújula interna y una externa se debe activar la casilla de Compass 1 y Compass 2 y presionar Start, luego se deberá girar el vehículo en distintas direcciones de tal modo que todas las caras del vehículo apunten hacia el suelo la calibración terminará cuando se complete por completo las barras verdes de Mag1 y Mag 2. En la sección de anexos encontrara de manera detallada la calibración del compass.

La calibración del compass proporciona datos fundamentales de la orientación con respecto al campo magnético de la tierra para mejorar la estabilidad del robot submarino. A continuación, se detalla los tipos de datos que proporciona el compass:

- **Campo magnético en tres ejes:**
  - Eje X: Mide la intensidad del campo magnético en dirección del eje X.
  - Eje Y: Mide la intensidad del campo magnético en dirección del eje Y.
  - Eje Z: Mide la intensidad del campo magnético en dirección del eje Z
- **Dirección magnética:** Combina datos con el GPS para indicar la dirección en la que apunta el robot submarino con respecto al norte magnético, siendo estos datos impórtate para la navegación y orientación del robot submarino.
- **Precisión de navegación:** Realizar y configurar correctamente la calibración del compass va a proporcionar datos con mayor precisión en la orientación del robot submarino, lo cual es esencial para mantener un rumbo estable y evitar desviaciones.
- **Mejora del control automático:** el sistema de control autónomo depende de los datos que proporcione el compass para poder ajustar la dirección y mantener estable el robot, el compass debe tener una calibración adecuada para mejorar la respuesta del sistema de control haciendo que el robot submarino sea más estable.
- **Reducción de derivas:** si el compass no se calibra de manera adecuada, robot podría sufrir desvíos que afecten a la estabilidad y precisión del sistema de



**Firmware:**

Como la versión del firmware inicial del robot submarino era la ArduSub 3.5.0 se hicieron algunas configuraciones para que permita cambiar de modo al robot con la incorporación del GPS, lo cual no permitió y solo se podía utilizar en modo manual, por lo que era necesario actualizar el firmware, pero surgió el inconveniente que no todas las versiones actualizadas de firmware son estables muchas son lanzadas en modo de prueba por lo que pueden presentar fallas.

**Solución:** Se decidió realizar pruebas con versiones estables de Firmware, lo cual asegura que estos funcionan correctamente hasta que se encontró la versión estable de ArduSub 4.0.1 fue la que permitió el cambio de modos en el robot submarino. A continuación, se mencionan algunas mejoras de la versión de firmware 4.0.1:

- Solución de problemas cuando se cambia de modo, causaba que el robot se sumergiera
- Soluciona el problema de armado de motores, no era posible armar motores después de realizar pruebas de motores.
- Se agregó nueva opción que permite detectar automáticamente la dirección de los propulsores, facilitando su configuración.
- Reintenta leer la ID de la brújula hasta 10 veces para evitar recalibración del compás.
- Mejoras en el comportamiento de retención de profundidad al alcanzar el fondo o superficie.
- Mejora en la integración con herramientas de planificación de misiones en modo AUTO, SURFACE, GUIDED Y CIRCLE.

**Problemas en la transmisión de video:**

Por la actualización del sistema operativo BlueOS al momento de conectar el robot a la estación de control este no presentaba la transmisión de video dentro del software QGroundControl.

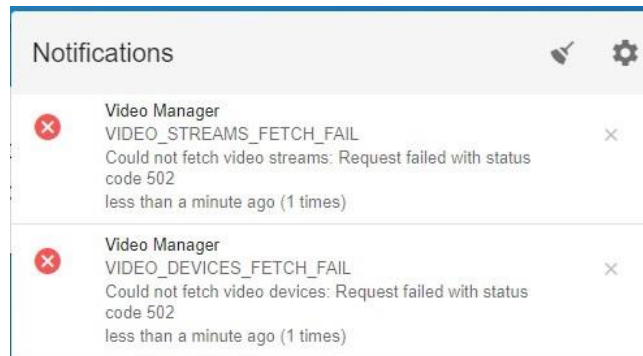


Figura 72: Notificaciones de problemas en la transmisión de video  
Imagen elaborada por el autor

**Solución:** Para solucionar este problema se usó la interfaz web que proporciona BlueOS. El acceso a la interfaz es por medio del navegador de su preferencia y escribiendo la siguiente dirección [192.168.2.2](http://192.168.2.2) o [blueos.local](http://blueos.local). Dentro de la interfaz se encontró notificaciones del problema en la transmisión de video y se pudo apreciar que no estaba agregada ninguna transmisión de video, por lo que se tuvo que agregar la ip de la Raspberry que es de donde se transmite el video. Para corregir este problema vaya a la opción **Video Streams** y seleccione **H264 USB Camera**, en la figura 73 se muestra la configuración para agregar la transmisión de video.



Figura 73: Configuración para habilitar la transmisión de video en la interfaz de BlueOS  
Imagen elaborada por el autor

## Navegación autónoma

La navegación autónoma se realizará a través del software Mission Planner y QGroundControl, se deberá tener buena recepción de los satélites del GPS y estar bien calibrado. En el siguiente diagrama se muestra la lógica del sistema de navegación autónoma por medio de puntos de referencia (Waypoint) que realizara el robot submarino.

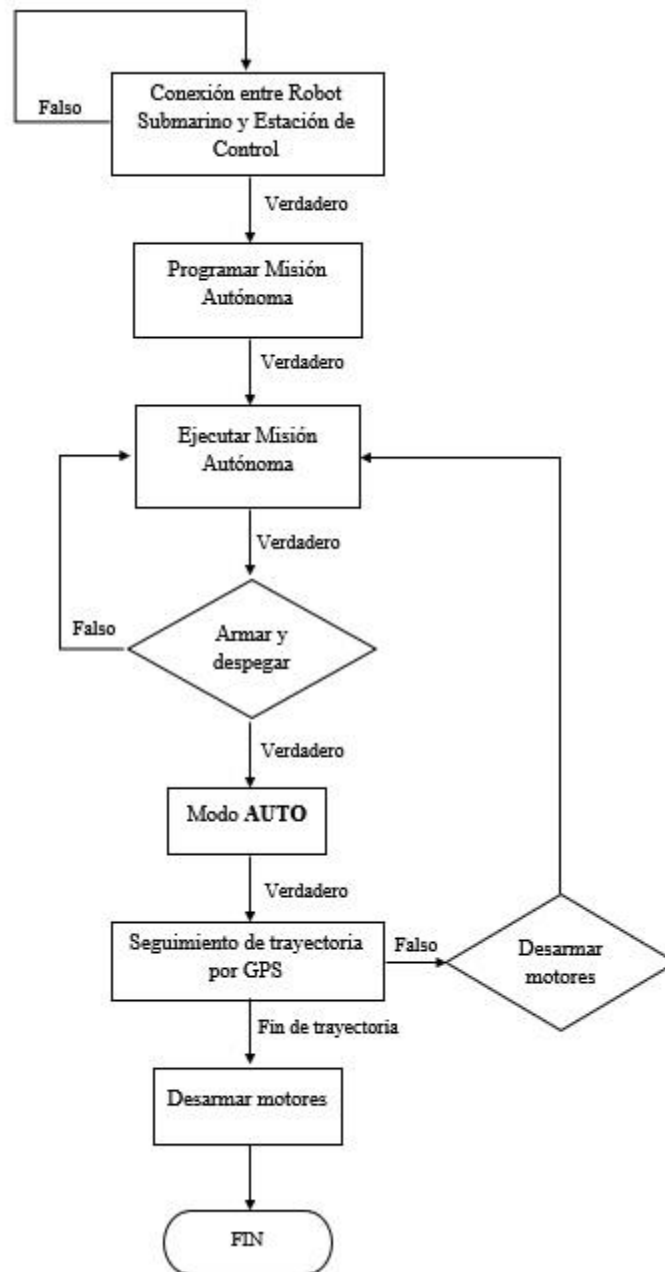


Figura 74: Diagrama de flujo de lógica de navegación autónoma  
Imagen elaborada por el autor

El proceso de control que realiza la Pixhawk se basa en una combinación de hardware y software que permite la navegación estable del vehículo.

### **Hardware**

- **Controlador de vuelo:** El controlador Pixhawk es el cerebro del sistema, se encarga de gestionar todos los sensores y actuadores.
- **Sensores:** La Pixhawk contiene sensores internos como el acelerómetro, magnetómetros y giroscopio para medir la orientación, dirección, velocidad y altitud, además permite incluir como el GPS que ayuda a establecer una posición de referencia.
- **Actuadores:** Los motores o servos ejecutan las órdenes del controlador de vuelo

### **Software**

- **Firmware:** El firmware de PX4 o ArduPilot contiene los algoritmos de control de vuelo y se instala en el controlador Pixhawk.
- **Software de estación de control en superficie:** Los softwares Mission Planner o QGroundControl se encargan de calibrar los sensores internos del Pixhawk, configurar parámetros, planificar misiones autónomas y monitorear el estado del vehículo.

A continuación, se explica con un esquema de sistema de control en la navegación autónoma de un vehículo utilizando el controlador Pixhawk y el software que hace de estación de control (QGroundControl o Mission Planner):

**Trayectoria (Mission Planner o QGroundControl):** El bloque de trayectoria representa la entrada deseada o trayectoria que el vehículo debe seguir. QGroundControl o Mission Planner son utilizados para planificar o diseñar la trayectoria que se desea que el robot siga.

**Sumador o Comparador:** Este símbolo representa la comparación entre la trayectoria real del robot con la trayectoria deseada.

**Controlador Pixhawk:** El Pixhawk es el cerebro del sistema, recibe el error desde el comparador y decide de qué manera ajustar los propulsores para minimizar algún error que imposibilite al robot llegar a la trayectoria deseada.

**Propulsores:** Los propulsores reciben las órdenes del controlador Pixhawk de ajuste de velocidad para modificar la posición y velocidad del robot, las ordenes que envía la



Pixhawk a los propulsores tienen como objetivo corregir algún error y aproximar al robot a la posición deseada.

**Sensores (Mission Planner o QGroundControl):** Representa la retroalimentación del sistema, en el cual incluye la información que envían los sensores sobre la orientación, dirección, posición y velocidad actual del robot, estos datos se envían al software que se hace de estación de control donde se comparan con la trayectoria deseada para continuar con el período de control.

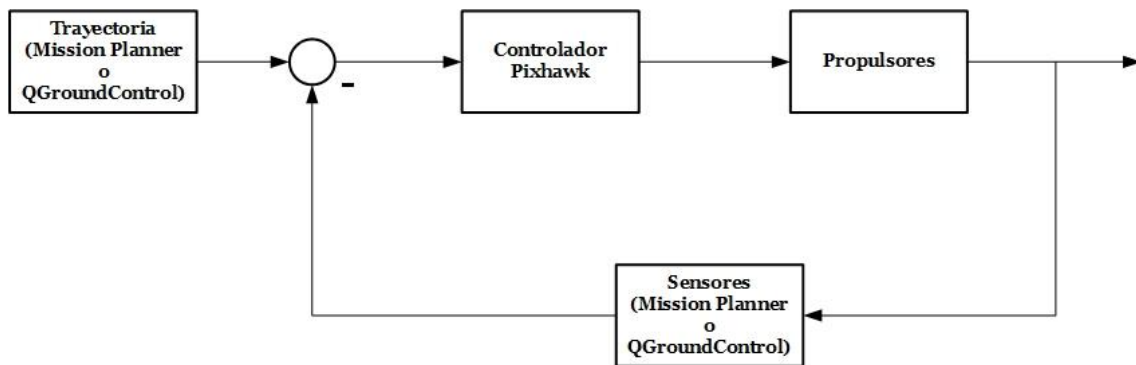


Figura 75: Esquema general del sistema de control

Imagen elaborada por el autor



Figura 76: Gráfica Roll Angular Rate

Imagen de la interfaz web de BlueOS

La figura 76 representa la tasa angular de balanceo (Roll Angular Rate) en grados por segundos (deg/s), medida que describe la velocidad angular, es decir representa la

velocidad a la que el robot se encuentra rotando alrededor del eje longitudinal Roll en función del tiempo.

**Roll Rate Estimated:** la curva de color naranja representa la tasa de balanceo estimada, donde se observa con fluctuación alrededor de la línea de referencia, lo que indica que el sistema está ajustando continuamente el balanceo.

**Roll Rate Setpoint:** la curva de color verde representa el valor de referencia a mantener, la proximidad entre la curva estimada y la curva de setpoint sugiere la precisión del control.

**Roll Rate Integral:** la curva de color celeste representa el valor integral de la tasa de balanceo entre -30 a 30 deg/s. este valor se puede relacionar con la acumulación de error a lo largo del tiempo, al presentar estabilización cercana a cero sugiere que el sistema está bien calibrado sin presentar grandes desviaciones acumuladas a lo largo del tiempo.

A pesar que la gráfica presenta fluctuaciones con variabilidad constante, lo que indica la existencia de perturbación externa o del propio comportamiento dinámico del sistema. A pesar de las fluctuaciones la tasa de balanceo estimada sigue cercana a el valor de referencia lo que indica que el sistema está realizando ajustes para mantener la estabilidad del robot indicando que el sistema está funcionando de manera correcta ajustando continuamente cualquier desviación para mantener la estabilidad anhelada.

## CAPITULO V

### 5. Pruebas y Resultados

#### 5.1. Resultado de entrenamiento del modelo Yolov4-tiny

El modelo personalizado entrenado con YOLOv4-tiny tardó menos de 2 horas en entrenar por completo, esto se debe a que es un modelo comprimido de YOLOv4 solo posee 29 capas convolucionales requieren menor complejidad computacional y menor tiempo de entrenamiento.

Para evaluar el rendimiento del modelo personalizado de detección de objetos de Estrella de Mar, tenemos las siguientes ecuaciones:

$$\text{Ecuación (1)} \quad \textit{Precision} = \frac{TP}{TP+FP}$$

$$\text{Ecuación (2)} \quad \textit{Recall} = \frac{TP}{TP+FN}$$

$$\text{Ecuación (3)} \quad mAP = \frac{1}{C} \sum_{K=i}^N P(k) \Delta R(k)$$

$$\text{Ecuación (4)} \quad F1 = \frac{2}{\frac{1}{\textit{Precision}} + \frac{1}{\textit{Recall}}}$$

La ecuación (1), es de Precisión; ecuación (2), de Recuperación; ecuación (3) Precisión media promedio mAP y la ecuación (4) puntaje F1. Donde tenemos que TP, es el número de objetos reconocidos correctamente; FP, el número de objetos reconocidos erróneamente; FN, el número de objetos no identificados; C, número de categorías de objetivo; N, número de umbral de pagares; K, umbral de pagares; P(k), precisión y R(k), recuperación.

Con los datos recopilados del entrenamiento de la red neuronal personalizada podremos realizar las gráficas de Precisión, Recall, mAP y F1, que nos permitirán conocer el rendimiento del modelo personalizado.

**Tabla 11. Resultados del entrenamiento de red neuronal convolucional personalizada.**

| Iteraciones | Precisión | Recall | mAP   | F1   |
|-------------|-----------|--------|-------|------|
| 1000        | 0,75      | 0,93   | 0,873 | 0,83 |
| 1500        | 0,84      | 1      | 0,97  | 0,91 |
| 2000        | 1         | 0,94   | 0,99  | 0,97 |
| 2500        | 0,94      | 0,94   | 0,99  | 0,94 |
| 3000        | 0,94      | 1      | 1     | 0,97 |
| 3500        | 0,94      | 1      | 1     | 0,97 |
| 4000        | 0,94      | 1      | 1     | 0,97 |
| 4500        | 0,94      | 1      | 1     | 0,97 |
| 5000        | 0,94      | 1      | 1     | 0,97 |
| 5500        | 0,94      | 1      | 1     | 0,97 |
| 6000        | 0,94      | 1      | 1     | 0,97 |

En el gráfico de precisión se puede ver que desde las 2500 iteraciones el modelo entrenado alcanza la estabilidad a 0,94 de precisión, lo que indica que el modelo entreno de manera correcta y que la posibilidad de que tenga presencia de falsos positivos al momento de poner a prueba el modelo es muy bajo.

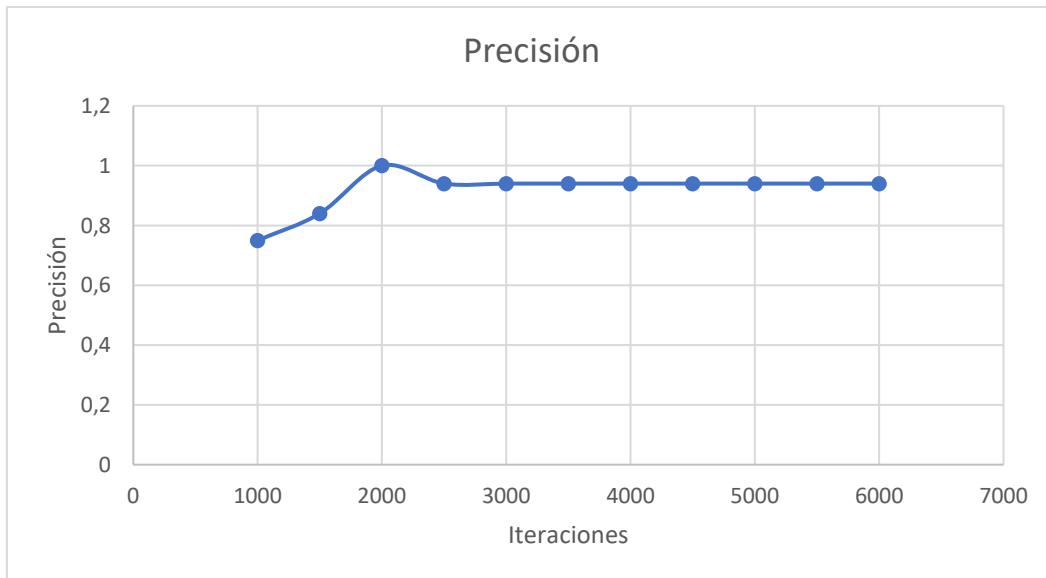


Figura 77: Gráfica de precisión del modelo entrenado

Imagen elaborada por el autor

En el gráfico de recuperación evalúa la correcta detección de verdaderos positivos entre todo los positivos reales, calculando la capacidad del modelo para detectar todos los objetos dentro de una imagen. Se aprecia que antes de las 3000 iteraciones el modelo presenta falsos negativos, es decir que no está detectando todos los verdaderos positivos

en la imagen, después de las 3000 iteraciones se logra la estabilidad de recuperación del modelo alcanzando un punto máximo de 1, lo que significaría que el modelo pasado las 3000 iteraciones ya no presenta falsos negativos y que la probabilidad de aparición de falsos positivos sea muy baja.

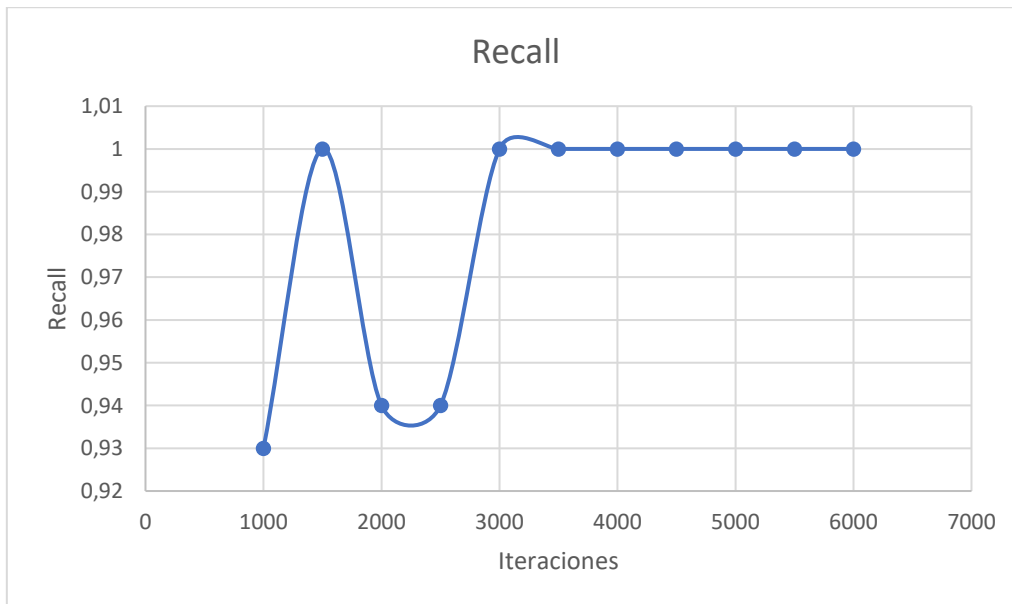


Figura 78: Gráfica de Recall del modelo entrenado

Imagen elaborada por el autor

La grafica de la precisión media promedio mAP empieza a alcanzar la estabilidad de precisión después de las 2000 iteraciones hasta que llega a las 3000 iteraciones donde alcanza un valor de 100% de precisión media, dándonos a entender que el modelo se ha entrenado de manera correcta.

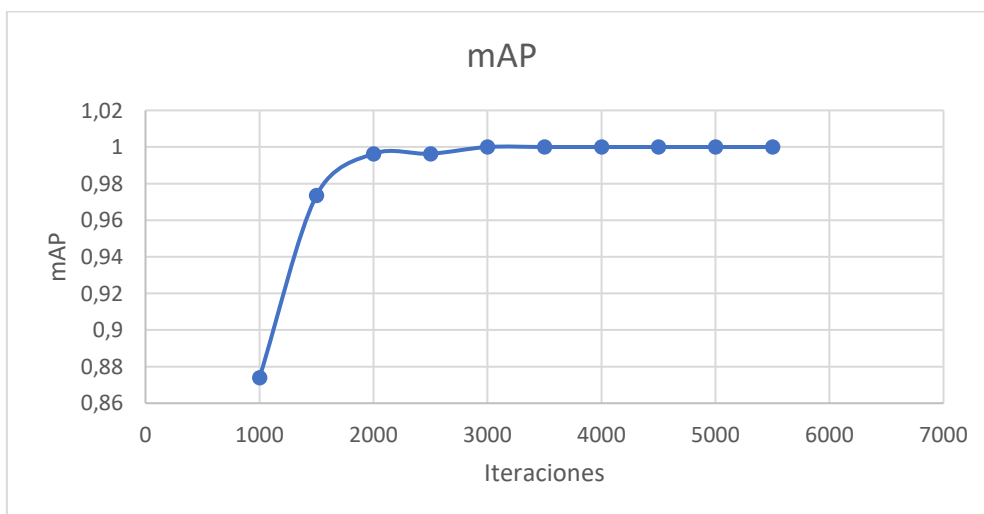


Figura 79: Gráfica de precisión media promedio mAP del modelo entrenado

Imagen elaborada por el autor

La grafica de puntuación F1 está basada en el promedio armónico de la Precisión y Recuperación (Recall), donde proporciona una evaluación equilibrada del rendimiento del modelo entrenado tomando en cuenta los falsos positivos y negativos del modelo. Se puede apreciar que alcanzada las 3000 iteraciones el modelo entrenado logra la estabilidad de promedio armónico tanto de precisión y recuperación.

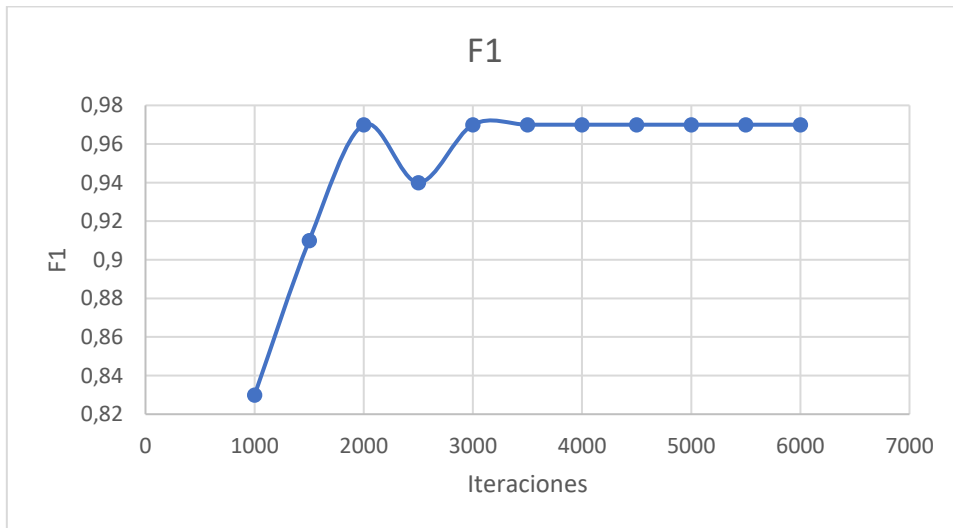


Figura 80: Gráfica de F1 del modelo entrenado

Imagen elaborada por el autor

En la gráfica de pérdida de modelo entrenado muestra cómo cambia la pérdida durante el entrenamiento del modelo, se puede apreciar que es una función decreciente alcanzo un valor bajo de pérdida de 0,48 antes de llegar a las 1000 iteraciones, lo que nos indica que el modelo está aprendiendo y mejorando su precisión. Después de las 2000 iteraciones el modelo alcanza una estabilidad de perdida indicando que el modelo ha alcanzado su punto máximo de mejoramiento y aprendizaje durante el entrenamiento.



Figura 81: Gráfica de Loss del modelo entrenado

Imagen elaborada por el autor

### 5.1.1. Pruebas del modelo entrenado

Ahora probaremos el código de detección de objetos y el modelo de red neuronal creado para el reconocimiento en tiempo real de la especie marina Estrella de Mar.

Como el modelo entrenado nos proporcionó weights o pesos desde las 1000 a 6000 iteraciones de entrenamiento probaremos el código y el modelo con los pesos de 1000, 3000 y 6000 iteraciones, esto debido al resultado arrojado por las gráficas nos dice que en las primeras 1000 iteraciones el modelo presenta cierta cantidad de falsos positivos pero aun así presenta un buen margen de precisión, en las 3000 iteraciones el modelo recién logra estabilizarse y en las 6000 iteraciones ya tendríamos un modelo con una precisión del 0,94%.

#### 1000

Cuando probamos el modelo entrenado con el peso o weights generado a las 1000 iteraciones nos encontramos que presenta falsos positivos en la ejecución del código de visión artificial.



Figura 82: Prueba del modelo entrenado con el peso o weights a 1000 iteraciones

Imagen elaborada por el autor



Figura 83: Presencia de falso positivo del modelo entrenado a 1000 iteraciones  
Imagen elaborada por el autor

### 3000

Con el peso o weights generado a las 3000 iteraciones aun presenta falsos positivos, esto se puede corregir agregando un if para que condicione que a partir de cierto porcentaje detecte la estrella de mar.

```
if conf >= [0.8286921381950378]:  
    cv2.imshow("Deteccion",img)
```



Figura 84: Prueba del modelo entrenado con el peso o weights a 3000 iteraciones  
Imagen elaborada por el autor



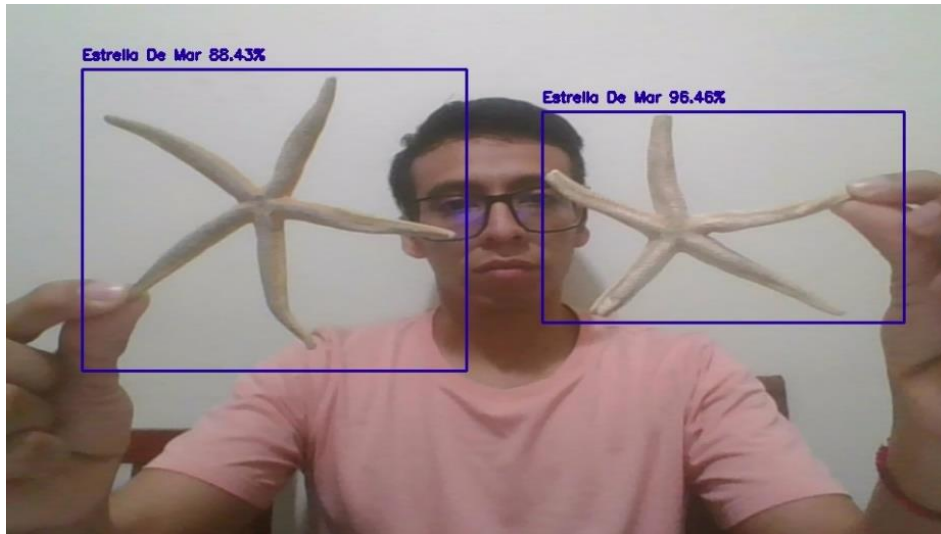


Figura 85: Corrección de presencia de falso positivo a 3000 iteraciones  
Imagen elaborada por el autor

## 6000

En las 6000 iteraciones el peso o weights generado logro la estabilización y no presenta falsos positivos en la ejecución del código de visión artificial, aunque aún se dejó la condición agregada cuando se probó el modelo en las 3000 iteraciones dado que el medio donde se probara es en el agua y no sabemos cómo responderá el modelo hasta ser probado en ese medio.



Figura 86: Prueba del modelo entrenado con el peso o weights a 6000 iteraciones  
Imagen elaborada por el autor

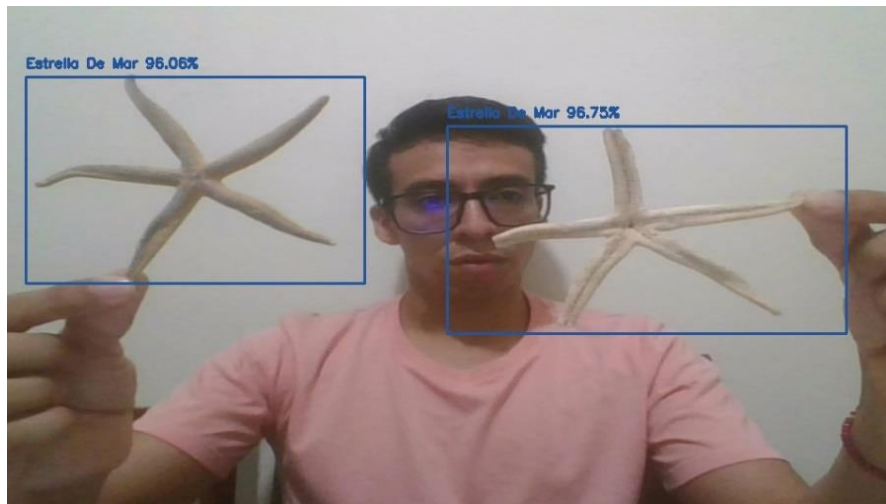


Figura 87: Prueba del modelo entrenado a 6000 iteración sin presencia de falsos positivos  
 Imagen elaborada por el autor

A continuación, se presenta una matriz de confusión de las pruebas realizadas en superficie la cual permitirá evaluar el desempeño del modelo de reconocimiento de la especie marina estrella de mar, la matriz de confusión facilita métricas sencillas para evaluar la exactitud y precisión del modelo. Las columnas representan las instancias reales y las filas representa las predicciones, es decir la matriz de confusión nos permite visualizar los aciertos y errores que está teniendo el modelo entrenado durante el proceso de reconocimiento de la estrella de mar en tiempo real.

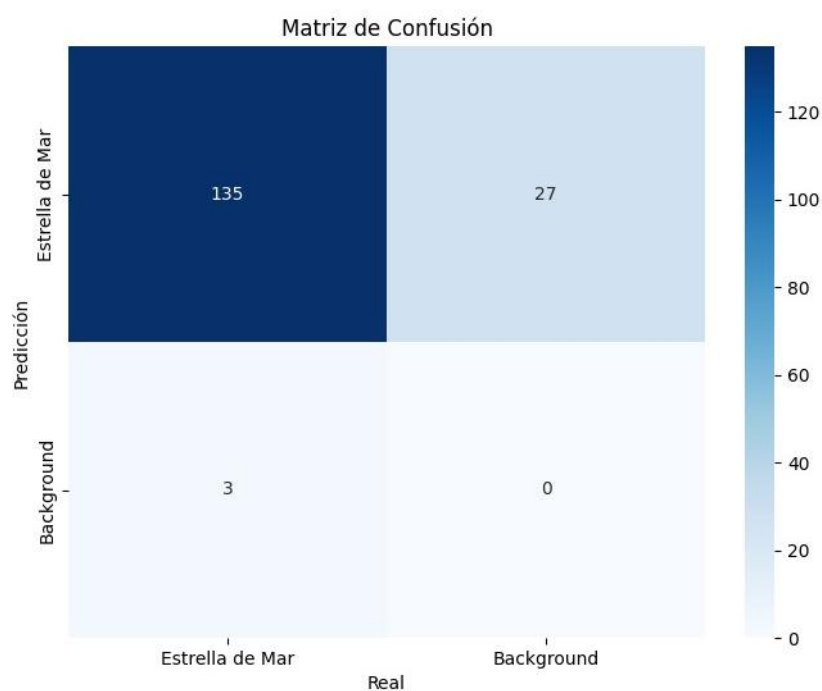


Figura 88: Matriz de confusión de pruebas realizadas al modelo entrenado en superficie  
 Imagen elaborada por el autor

La matriz de confusión presenta 165 muestras reales que son estrellas de mar (clase positiva), donde tenemos que 135 son predicciones correctas, 27 son predicciones incorrectas, 3 son números de objetos no identificados y 0 son verdaderos negativos en este caso no se realiza predicciones que indiquen que un determinado objeto que aparezca ante la cámara no es una estrella de mar.

Con los datos obtenidos de la matriz de confusión podemos obtener las siguientes métricas de evaluación:

**Exactitud:** La exactitud del modelo nos indica la proporción de predicciones correctas tanto de verdaderos positivos como negativos sobre el total de predicciones, donde tenemos una exactitud de 81.81% un valor alto para pruebas realizadas entre weights de 1000, 3000 y 6000 iteraciones.

se puede calcular si se conocen todas las predicciones correctas e incorrectas. En este caso con la información disponible, la exactitud sería:

$$Exactitud = \frac{TP + TN}{Total\ de\ Predicciones} = \frac{135 + 0}{165} = 0.8181\ (81.81\%)$$

**Precisión:** La precisión general del modelo en las pruebas realizadas en superficie con los weights 1000, 3000 y 6000 iteraciones, arrojaron un valor de 81.81%, lo que indica que el modelo realiza la mayoría de predicciones de manera correcta.

$$Precisión = \frac{TP}{TP + FP} = \frac{135}{135 + 30} = 0.8181\ (81.81\%)$$

**Sensibilidad:** La sensibilidad que posee el modelo para identificar correctamente las instancias positiva, es considerablemente alta con un 97.82% entre los 3 weights evaluados.

$$Sensibilidad = \frac{TP}{TP + FN} = \frac{135}{135 + 3} = 0.9782\ (97.82\%)$$

De acuerdo a los datos arrojados por la matriz de confusión se puede decir que el modelo tiene alta precisión, indicando que cuando predice Estrella de Mar es muy probable que la predicción sea correcta.

### 5.1.2. Pruebas del modelo bajo el agua

Para las pruebas del modelo entrenado bajo el agua se utilizó el weights generado a las 6000 iteraciones, ya que nos arrojó mejores resultados de detección de la estrella de mar en las pruebas realizadas en superficie.

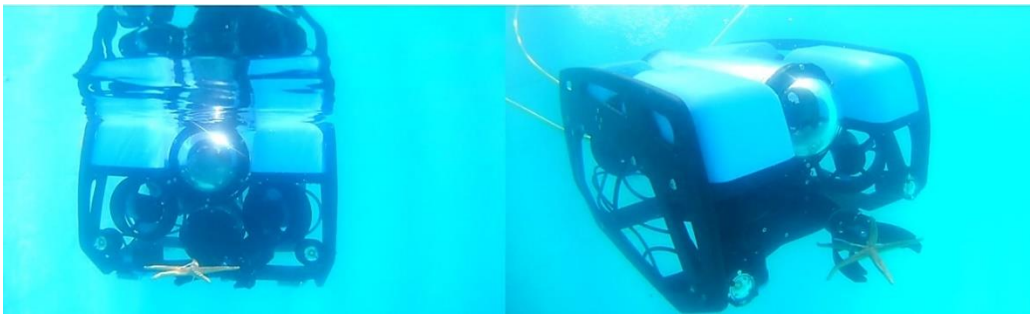


Figura 89: Pruebas del modelo entrenado bajo el agua con el robot submarino  
Imagen elaborada por el autor

Durante el reconocimiento de la estrella de mar el robot submarino tendrá que mantenerse estable para tomar buenas capturas durante las pruebas. Con la ayuda del software QGroundControl podremos poner el robot en modo estable o position hold y que el proceso de detección de la estrella de mar se pueda ejecutar sin inconvenientes.

Las pruebas bajo el agua nos arrojaron resultados de precisión desde 52.18% hasta 90.11% sin la presencia de falsos positivos. Cabe aclarar que mostro resultados de precisión más bajo cuando se manipulaba la posición del robot para verificar si mantenía su posición.

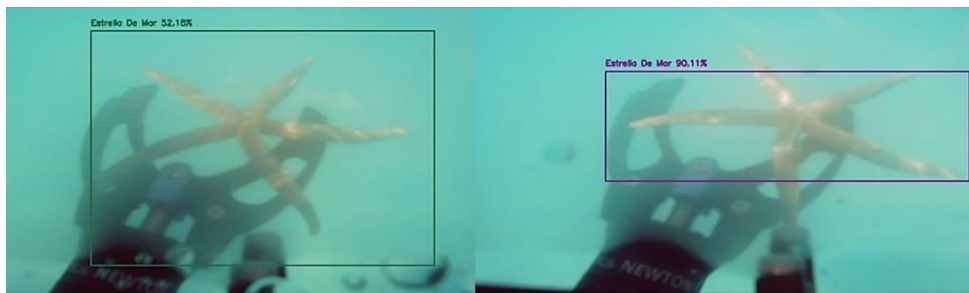


Figura 90: Resultados de precisión entre 52.18% hasta 90.11% sin presencia de falsos positivos  
Imagen elaborada por el autor

En la figura 91 se puede observar el comportamiento de estabilidad del robot submarino cuando presenta alguna perturbación en Pitch/Roll que desestabilice al robot, a la vez se observa como Pitch y Roll se manejan de manera automática con el fin de hacer que el robot se mantenga estable y pueda realizar buenas capturas de imágenes del proceso de detección de la estrella de mar.

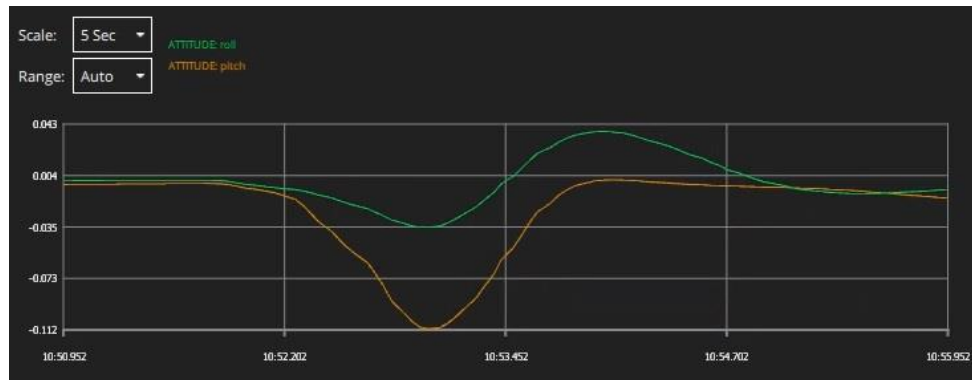


Figura 91: Gráfica de Roll y Pitch en modo estable  
Imagen elaborada por el autor

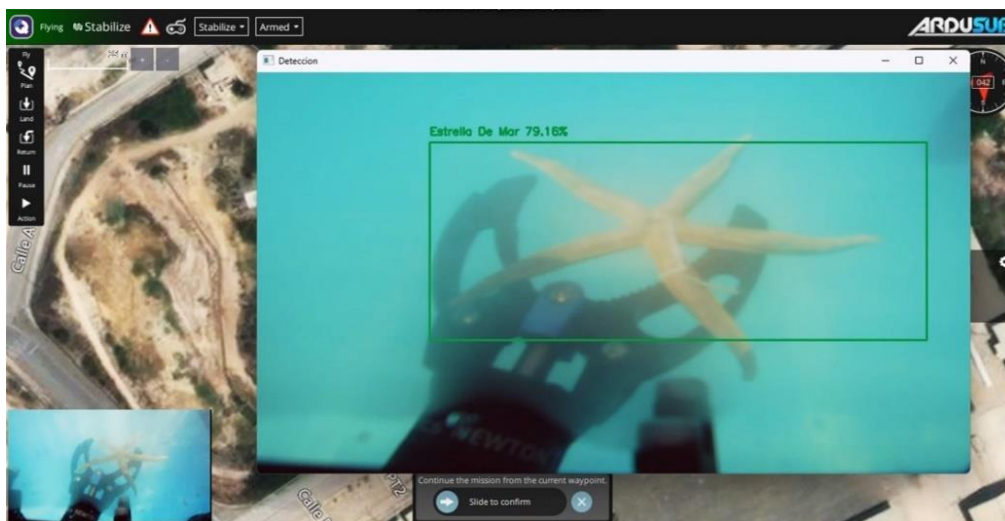


Figura 92: Pruebas del modelo entrenado con el robot submarino en modo estable  
Imagen elaborada por el autor

Durante la prueba de estabilidad del robot la estrella de mar se desplazó de su ubicación, pero aun así el modelo entrenado reconoció a la estrella de mar, pero presentando resultado de precisión por debajo de 59.36%



Figura 93: Desplazamiento de la estrella de mar durante pruebas presenta una precisión de 59.36%

Imagen elaborada por el autor

También se probó el modelo de detección de la estrella de mar durante la noche, el cual presento buenos resultados llegando a tener una precisión de 97.43% sin realizar perturbación al robot submarino durante el proceso de detección.



Figura 94: Pruebas del modelo entrenado bajo el agua en la noche con el robot submarino

Imagen elaborada por el autor

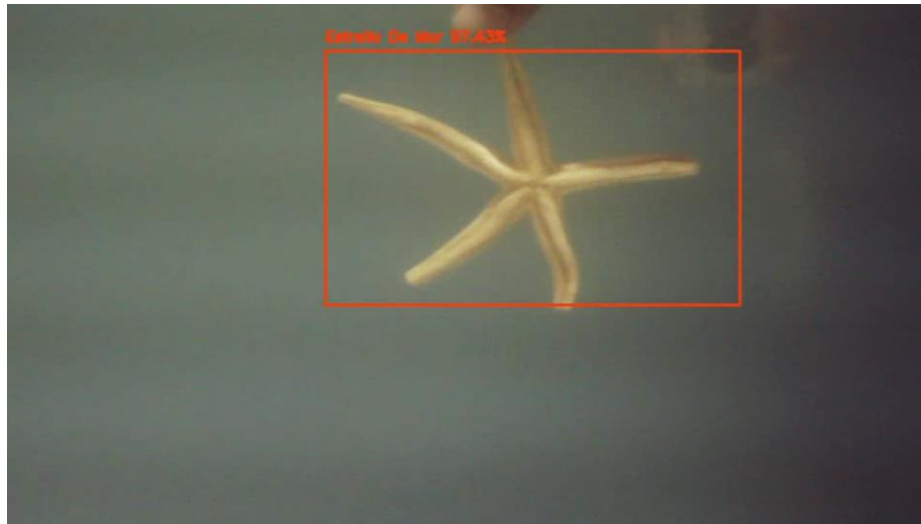


Figura 95: Precisión de 97.43% del modelo entrenado durante las pruebas realizadas en la noche  
 Imagen elaborada por el autor

A continuación, se presenta la matriz de confusión de las pruebas realizadas bajo el agua utilizando el weights generado a las 6000 iteraciones. La matriz de confusión presenta 264 muestras reales que son estrellas de mar, donde tenemos 239 predicciones correctas, 17 son predicciones incorrectas, 8 son números de objetos no identificados.

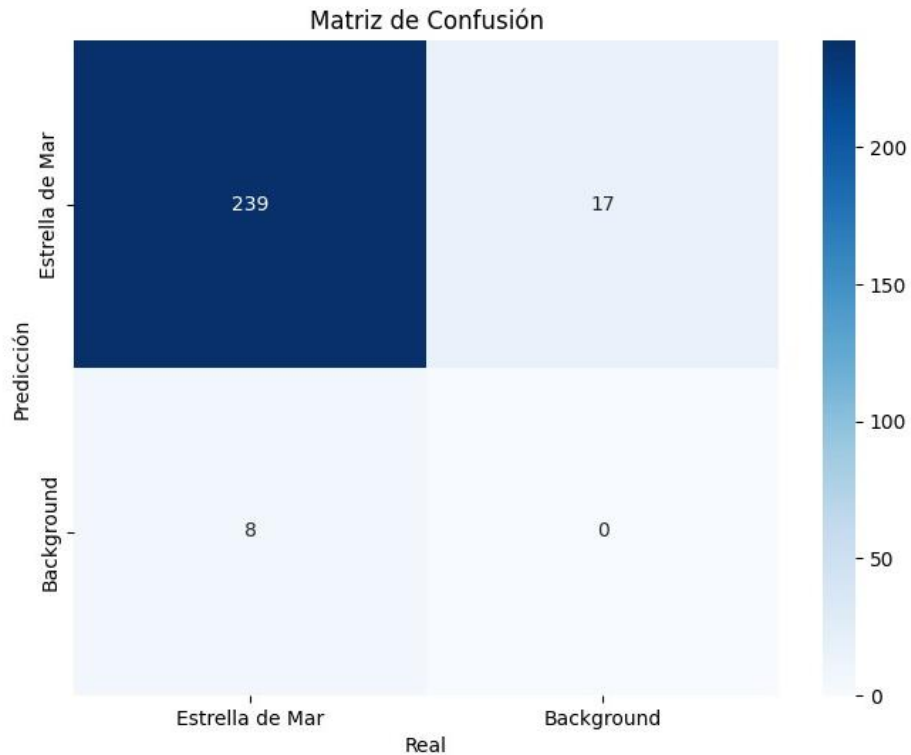


Figura 96: Matriz de confusión de pruebas realizadas al modelo entrenado bajo el agua  
 Imagen elaborada por el autor

Con los datos obtenidos de la matriz de confusión de las pruebas bajo el agua podemos obtener las siguientes métricas de evaluación:

Exactitud: En exactitud de las pruebas realizadas bajo el agua nos indica que el modelo realizó 90.53% predicciones correctas.

$$\text{Exactitud} = \frac{TP + TN}{\text{Total de Predicciones}} = \frac{239 + 0}{264} = 0.9053 \text{ (90.53\%)}$$

Precisión: La precisión del modelo en las pruebas realizadas bajo el agua arrojaron un valor de 93.35%, demostrando que el modelo realiza la mayoría de predicciones de manera correcta.

$$\text{Precisión} = \frac{TP}{TP + FP} = \frac{239}{239 + 17} = 0.9335 \text{ (93.35\%)}$$

Sensibilidad: La sensibilidad modelo para identificar correctamente la especie marina estrella de mar positiva, es considerablemente alta con un 96.76%.

$$\text{Sensibilidad} = \frac{TP}{TP + FN} = \frac{239}{239 + 8} = 0.9676 \text{ (96.76\%)}$$

A pesar que el modelo de reconocimiento de la especie marina estrella de mar en pruebas realizadas bajo el agua presenta 17 predicciones incorrectas y 8 estrellas de mar no identificadas realizó un buen proceso de reconocimiento ya que cuenta con 239 predicciones correctas de la estrella de mar.

De acuerdo a las pruebas realizadas se puede afirmar que el modelo entrenado con YOLOv4-Tiny ha obtenido buenos resultados de precisión tanto en superficie como bajo el agua e incluso en la noche donde la luz solo proviene de la iluminación instalada en el robot submarino.





Figura 97: Predicciones erróneas realizadas bajo el agua por el modelo entrenado  
Imagen elaborada por el autor



Figura 98: presencia de falsos negativos en pruebas realizadas bajo el agua  
Imagen elaborada por el autor

## 5.2. Resultado del sistema de navegación autónoma

Antes de realizar las pruebas de las misiones autónomas programadas se realizaron pruebas en modo manual para comprobar que los motores funcionen correctamente y que giren en el sentido correcto para que el robot submarino pueda avanzar, retroceder, descender y ascender.

Cuando se procedió a realizar maniobras con el joystick en modo manual nos encontramos con la novedad que el robot submarino no avanzaba, no retrocedía y al momento de querer descender el robot se hacía un lado, por tal motivo se tuvo que volver a calibrar el acelerómetro, compass y verificar que el giro de los motores sea en el sentido correcto. Terminada las calibraciones del acelerómetro, el compass y teniendo corregido el giro de los motores, se pudo probar de manera correcta el avance, retroceso, descenso y ascenso del robot



Figura 99: Maniobras del robot submarino con el joystick en modo manual

Imagen elaborada por el autor

Luego de haber verificado el correcto funcionamiento de los motores del robot submarino y que este pueda desplazarse sin problemas de dirección, podremos realizar las pruebas de las misiones autónomas programadas.

### Prueba 1

La prueba número uno de navegación autónoma se realizó con Mission Planner, donde trazamos una trayectoria en línea recta que seguirá el robot submarino por medio de Waypoint, por motivos de espacio en la piscina.



Figura 100: Prueba 1 de misión autónoma realizada en Mission Planner

Imagen elaborada por el autor

Como resultado de la misión se puede decir que fue exitosa y no hubo problemas con el cumplimiento de la ruta programada, como se muestra en la figura 101 se aprecia que el robot submarino completo la ruta programada por el sistema de modo automático.



Figura 101: Trayectoria trazada de la misión autónoma realizada en Mission Planner

Imagen elaborada por el autor

En la figura 102 se muestra el comportamiento del robot submarino durante la misión autónoma, en cuanto a la respuesta de Roll y Pitch se observa que se maneja de manera automática, la gráfica de Pitch (azul) tiene una ligera inclinación la cual nos indica que el robot submarino se encuentra sumergido, mientras las demás salidas tratan de mantener la posición con respecto a la horizontal del vehículo que en este caso se encuentra ligeramente inclinada hacia abajo.

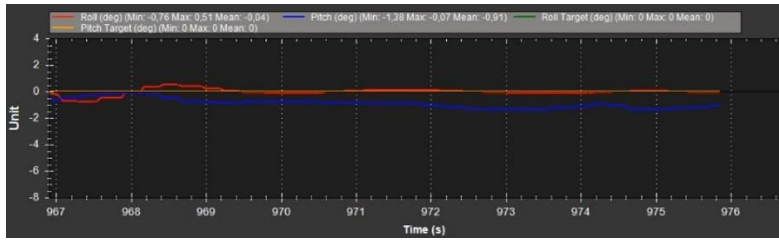


Figura 102: Gráfica de Roll y Pitch durante la misión autónoma realizada en Mission Planner

Imagen elaborada por el autor

## Prueba 2

En la prueba número dos utilizamos el software QGroundControl como estación de control y trazamos una ruta recta la cual seguirá el robot submarino por medio de puntos de referencia o Waypoints.

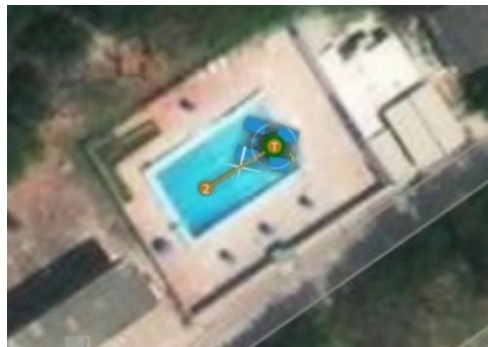


Figura 103: Prueba 2 de misión autónoma realizada en QGroundControl

Imagen elaborada por el autor

Los resultados obtenidos luego de que el robot submarino realizara la misión autónoma nos da a entender que se realizó sin inconvenientes, en la siguiente figura 104 se puede observar como el robot cumple con la misión autónoma, dejando un rastro de trayectoria confirmando que el robot se trasladó de un punto a otro.

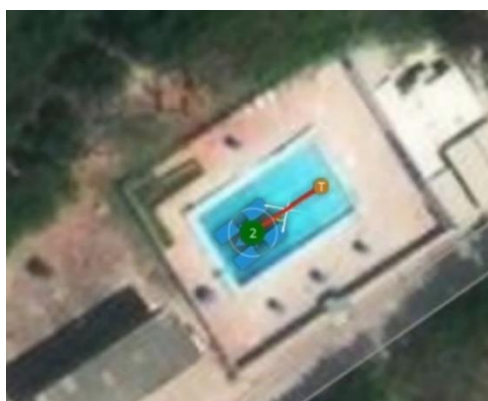


Figura 104: Trayectoria trazada de la misión autónoma realizada en QGroundControl

Imagen elaborada por el autor

En la prueba numero dos solo se muestra el comportamiento del robot submarino en la salida Pitch durante la misión autónoma, donde se observa que al momento de empezar la misión este presenta un cambio repentino hacia abajo, dando a entender que es el momento en que el robot empieza a descender. En la figura 105 se muestra que como el robot empieza el ascenso al finalizar la misión autónoma.

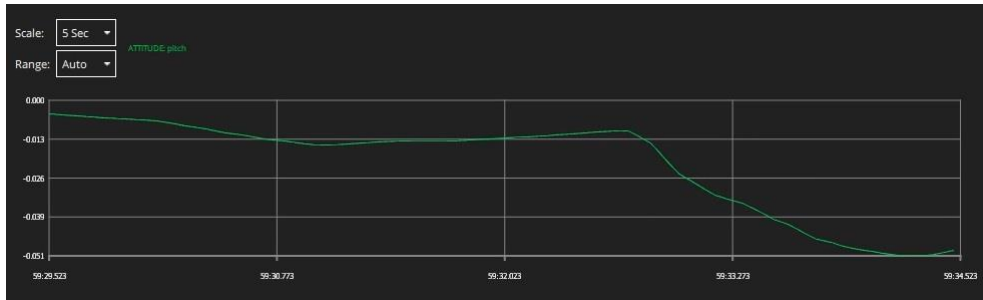


Figura 105: Gráfica del comportamiento de Pitch durante la misión autónoma

Imagen elaborada por el autor

### Prueba 3

La prueba número tres se volvió a realizar en el software QGroundControl en el cual se programó una ruta con cuatro puntos de referencia o Waypoints donde el robot submarino seguirá la trayectoria mostrada en la figura 106.



Figura 106: Prueba 3 de misión autónoma realizada en QGroundControl

Imagen elaborada por el autor

Terminada la prueba de la misión autónoma con 4 puntos de referencia, se puede apreciar en la figura 107 que en la trayectoria marcada por QGroundControl muestra que al parecer el robot no llegó exactamente al Waypoint 2 destinado, pero eso no quiere decir que la misión autónoma no haya sido exitosa ya que el programa pudo haber marcado el giro dentro del rango de ese Waypoint.

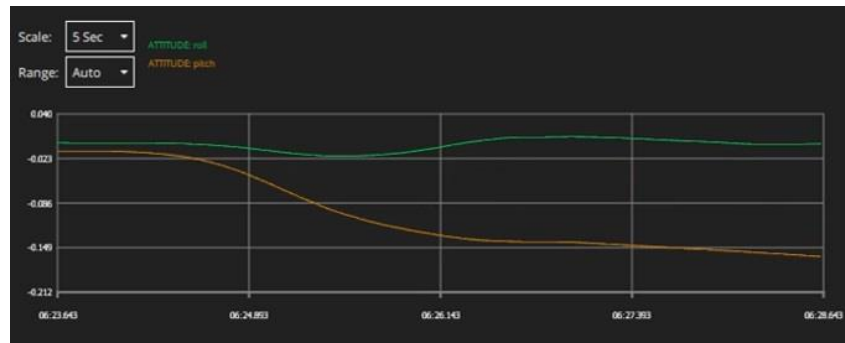


Figura 107: Trayectoria trazada de la prueba 3 de misión autónoma realizada en QGroundControl  
Imagen elaborada por el autor

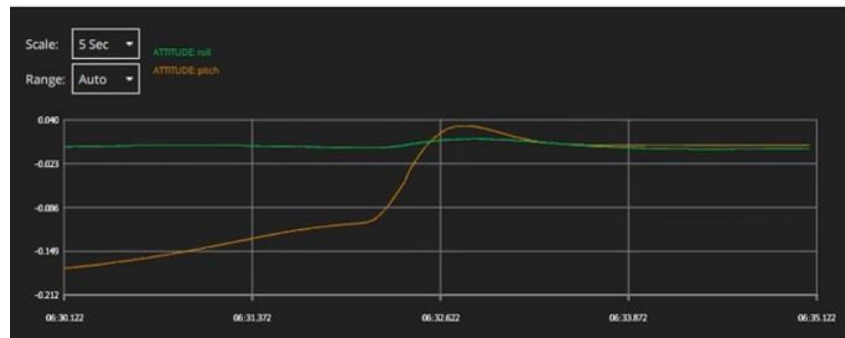
En la figura 108 se puede observar como el robot submarino si llega por completo al Waypoint 2 y que el QGroundControl marco el giro dentro del rango estimado en este punto, si se observa con la figura en el Waypoint 2 este se encuentra rodeado por un círculo blanco el cual marca el rango de giro de ese Waypoint. Analizado con detenimiento este detalle se puede decir que el robot submarino cumplió exitosamente la misión autónoma programada.



Figura 108: Comprobación de cumplimiento de trayectoria en el Waypoint 2  
Imagen elaborada por el autor



(a)



(b)

Figura 109: Gráfica de comportamiento de Roll y Pitch durante la misión autónoma realizada en QGroundControl

Imagen elaborada por el autor

En la figura 109 (a) se muestra los valores de Roll y Pitch en la que el valor de Pitch (naranja) empieza a descender durante la misión autónoma programada, mientras que el valor de Roll (verde) mantiene la posición del vehículo con respecto a la horizontal. Al llegar al final de la misión autónoma la figura 109 (b) muestra el momento del ascenso del robot submarino.

La siguiente gráfica representa una comparación tridimensional entre trayectoria esperada y la trayectoria real del robot submarino.

Trayectoria esperada: La línea azul representa el recorrido ideal o planificado para el robot submarino, en el cual se espera que el robot siga la trayectoria de manera controlada sin desviaciones significativas durante la navegación del robot.

Trayectoria real: La línea roja describe la trayectoria en condiciones reales del robot submarino, en la cual se puede apreciar pequeñas variaciones debido a los sensores del Pixhawk como el acelerómetro, giroscopio, brújula y GPS trabajan conjuntamente para seguir la trayectoria planificada, este tipo de desviaciones son comunes debido a la dinámica del entorno en el que se encuentre el robot.

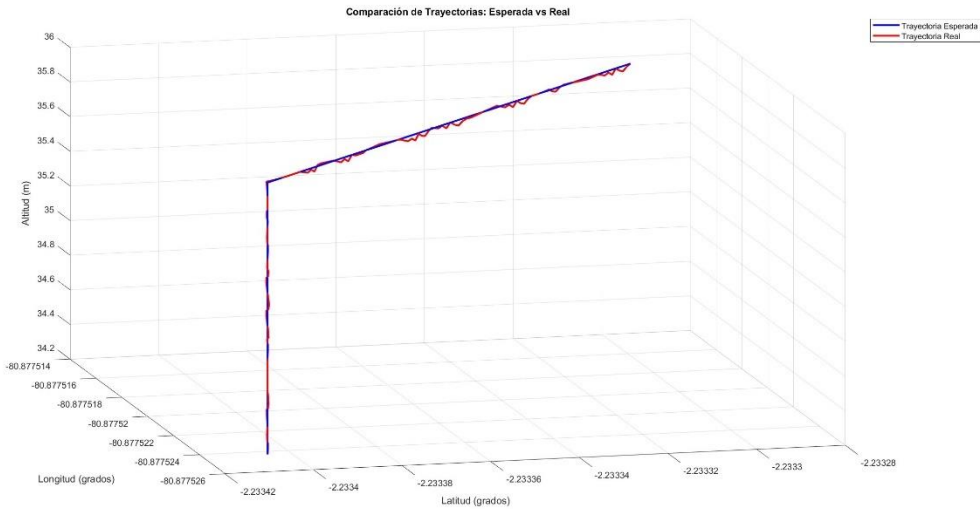


Figura 110: Gráfica tridimensional entre la trayectoria esperada vs la trayectoria real  
Imagen elaborada por el autor



Figura 111: Gráfica del robot submarino durante la trayectoria  
Imagen elaborada por el autor

La comparación entre la trayectoria esperada y la trayectoria real permite identificar el error entre las dos trayectorias. Para evaluar el sistema de navegación del robot submarino se calculó el error cuadrático medio.

La fórmula del error cuadrático medio

$$ECM = \frac{1}{n} \sum_{i=1}^n (x_{i\text{real}} - x_{i\text{esperado}})^2$$

$n$  representa el número total de muestras o de datos

$x_{i\text{real}}$  es el valor real de la trayectoria

$x_{i\text{esperado}}$  es el valor esperado de trayectoria



Error Cuadrático Medio (X): 0.000000

Error Cuadrático Medio (Y): 0.000000

Error Cuadrático Medio (Z): 0.000082

Error Cuadrático Medio Total: 0.000082

$$ECM = \frac{1}{n} \sum_{i=1}^n (x_i^{real} - x_i^{esperado})^2 = 0.000000017485$$

$$ECM = \frac{1}{n} \sum_{i=1}^n (y_i^{real} - y_i^{esperado})^2 = 0.0000001969$$

$$ECM = \frac{1}{n} \sum_{i=1}^n (z_i^{real} - z_i^{esperado})^2 = 0.000082$$

El error cuadrático medio total engloba las tres coordenadas que sería la siguiente:

$$ECM = \frac{1}{n} \sum_{i=1}^n [(x_i^{real} - x_i^{esperado})^2 + (y_i^{real} - y_i^{esperado})^2 + (z_i^{real} - z_i^{esperado})^2] = 0.000082$$

La siguiente gráfica muestra el error cuadrático medio entre la trayectoria esperada y la trayectoria real en las coordenadas X, Y y Z.

ECM en la coordenada X: En la coordenada X el error es muy pequeño de  $10^{-8}$ , indicando que el control en esta coordenada es preciso, aunque a lo largo de los datos presenta picos al final se debe a pequeñas correcciones durante la trayectoria.

ECM en la coordenada Y: El error en la coordenada Y es a penas mayor que en la coordenada en X de  $10^{-7}$ , mostrando un patrón de ondas debido al que el sistema trata de seguir la trayectoria planificada ajustando constantemente los sensores del controlador Pixhawk para seguir la misión planificada

ECM en la coordenada Z: La coordenada Z presenta un error mayor de 0.0025, con picos pronunciados al inicio del descenso del robot debido al cambio brusco que presenta al cambio de giro de los motores para que el robot empiece la etapa de descenso.

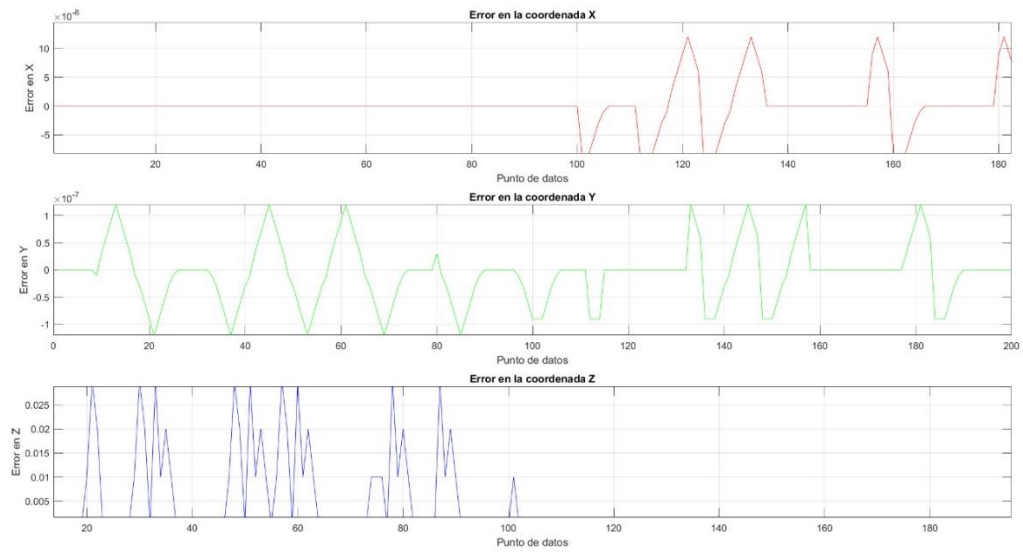


Figura 112: Gráfica del Error Cuadrático medio en coordenadas X, Y y Z  
 Imagen elaborada por el autor

## CONCLUSIONES

El estudio de los sistemas eléctricos que componen al robot submarino de BlueRobotics, fue de gran ayuda para empezar a comprender el funcionamiento del robot submarino, de cómo se encuentran conectados los motores, como se establece la comunicación entre la placa Pixhawk y la Raspberry, también como se establece la comunicación entre el robot y la estación de control, siendo la tarjeta Fathom-X un componente indispensable para que esta comunicación ocurra. Al comprender el funcionamiento electrónico del robot sirvió para establecer una solución al diseño del sistema de navegación autónoma e implementar un sistema de posicionamiento geográfico como el GPS.

Para comprobar que el modelo entrenado para que realice el reconocimiento de la especie marina estrella de mar se utilizó la matriz de confusión la cual proporciona una visión clara del número de predicciones correctas e incorrectas y permite calcular métricas como precisión, recall y exactitud que permiten entender la eficiencia del modelo. La matriz de confusión arrojó un gran desempeño a la hora de identificar la estrella de mar con una precisión de 93.35%, con recall o sensibilidad con 96.76% y exactitud de 90.53% en pruebas realizadas bajo el agua con una muestra de 264, donde 239 fueron predicciones correctas (verdaderos positivos), 17 predicciones incorrectas (falsos positivos) y 8 fueron objetos no identificados (falsos negativos), lo que significa que el modelo predice correctamente la mayoría de las estrellas de mar y que muy pocas veces presenta falsos positivos y negativos.

La evaluación de la trayectoria se realizó por medio del cálculo del error cuadrático medio donde se compara la trayectoria ideal con la trayectoria real realizada por el robot submarino, las desviaciones entre las dos trayectorias permite identificar el error cuadrático medio lo cual fue crucial para identificar los retos a los que se enfrenta el robot y a evaluar la efectividad del sistema de navegación del robot submarino, donde se obtuvo un error cuadrático medio en la coordenada X muy pequeño de 0.000000017485, indicando que el sistema de navegación en esta coordenada es bastante preciso a lo largo de la trayectoria. El error en la coordenada Y es un poco mayor con respecto a X, pero sigue siendo un valor pequeño de 0.0000001969, donde se muestra un pequeño patrón de ondas que indican que los sensores integrados al Pixhawk como el acelerómetro, giroscopio, brújula y GPS trabajan conjuntamente para seguir y corregir la trayectoria planificada. El error en la coordenada Z presenta un error de 0.000082, donde presenta

perturbaciones al inicio del descenso debido al cambio brusco del giro de los motores para empezar a descender.

El sistema de navegación ha sido diseñado para que se programe misiones autónomas siguiendo diferentes puntos de referencia o marcando una trayectoria específica mediante código de programación. El uso de los softwares Mission Planner y QGround Control facilito la configuración de parámetros importantes y necesarios para el robot como la calibración del acelerómetro el compass interno y externo GPS, configurar parámetros para que el GPS sea incluido en el EKF (filtro de Kalman extendido) y a la programación de misiones autónomas por medio de puntos de referencia o Waypoint. Aunque los dos softwares Mission Planner y QGround Control permiten trabajar con el robot submarino, el mejor software para trabajar con robots submarinos es QGroundControl ya que proporciona una interfaz sencilla, la programación de misiones autónomas es de manera sencilla, proporciona una pequeña interfaz donde visualiza el entorno bajo el agua es decir lo que va viendo el robot submarino a través de la cámara incorporada y también permite que el robot pueda ser controlado en modo manual.

A medida que el número de iteraciones aumenta, la red neuronal convolucional que fue entrenada muestra una mejora constante en las métricas de evaluación como precisión, Recall, mAP y F1. La precisión del modelo entrenado paso de 75% a 94% de precisión indicando que el modelo se vuelve más preciso a medida que va aumentando el número de iteraciones durante el entrenamiento. La métrica de Recall durante el entrenamiento mantiene valores altos de 93% a 100% lo que indica que el modelo entrenado captura la mayoría de objetos presentes en la imagen. En la precisión media promedio mAP aumenta de 87.3% a 100% lo que significa una mejora en la precisión promedio de detección. En la puntuación F1 mejora de 83% a 97% indicando un buen equilibrio entre el promedio armónico entre precisión y recall.

## RECOMENDACIONES

Si se desea aplicar el sistema de visión artificial desarrollado en esta propuesta y aplicarlo a algún robot, se recomienda reemplazar el uso de la Raspberry Pi por la Jetson Nano la cual posee una GPU 128-core Maxwell, mucho más potente ideal para aplicaciones de inteligencia artificial y visión artificial para robótica avanzada evitando que exista retraso en la detección del objeto. En el caso de este proyecto la Jetson Nano se adapta muy bien al sistema operativo BlueOS funcionando como intermediario entre el controlador Pixhawk y la estación de control, sin que sobre esfuercen los recursos computacionales que posee.

Para futuros trabajos con el robot submarino se recomienda implementar un sistema de control de estabilidad robusto integrando herramientas de medición de posición de gran precisión y utilizando un control PID avanzado que sea capaz de ajustar parámetros en función de las condiciones en que se encuentre el robot para buscar minimizar el error en el sistema de control de estabilidad y que se garantice el correcto funcionamiento del robot en condiciones submarinas adversas como el cambio de mareas y al constante oleaje que presenta el mar.

El robot submarino con el que se trabajó en este proyecto cuenta con herramientas que no fueron utilizadas como el Gripper y Ping360 y que se podría aprovechar para futuros trabajos como es el caso del Ping360, es un sonar que está diseñado para navegar en entornos de baja visibilidad el cual ayuda a detectar obstáculos en tiempo real. También es adecuado para mapear entornos bajo el agua generando un mapa 2D del entorno en que se encuentra, otra aplicación del Ping360 sería el seguimiento de trayectoria en combinación con un sistema de control de estabilidad robusto que permitirá que el robot sea capaz de mantener una trayectoria o misión planificada, además se podría implementar para el seguimiento y localización de objetos, permitiendo que el robot sea capaz de detectar y posicionarse en relación al objeto a identificar y aprovechando el Gripper capturar al objeto.

Para compensar la confiabilidad del sistema de reconocimiento de imagen, se debe realizar el entrenamiento de la red neuronal con más de 500 imágenes ya que entre más imágenes tenga el entrenamiento la red neuronal aprenderá más característica del objeto a detectar y por ende el modelo entrenado alcanzaría grandes resultados en cuanto a

precisión de detección. Además, se recomienda realizar el entrenamiento de la red neuronal en una máquina que cuente con GPU para que el proceso de entrenamiento sea mucho más rápido que si se realiza con una máquina que solo cuenta con CPU. Si no se cuenta con GPU se recomienda realizar el entrenamiento en Google Colab ya que proporciona memoria GPU gratuita, además no se recomienda que el entrenamiento se realice en la Raspberry ya que al poseer poca capacidad computacional durante el entrenamiento podría presentar sobrecalentamiento excesivo ocasionando que la Raspberry se apague.

## BIBLIOGRAFÍA

- [1] H. A. Moreno, R. Saltarén, L. Puglisi, I. Carrera, P. Cárdenas y C. Álvarez, «Robótica Submarina: Conceptos, Elementos, Modelado y Control,» *Elsevier*, vol. 11, nº 1, pp. 3-19, enero 2014.
- [2] J. Bellingham, *Plataformas: Vehículos Submarinos Autónomos*, Segunda Edición ed., Academic Press, 2009.
- [3] E. Lucín Recalde, *Implementación de un Prototipo Sumergible Híbrido para adquirir dato de Temperatura e Imágenes que ayude como herramienta de medición para los estudiantes de la Facultad de Ciencias del Mar*[Tesis de grado de Ingeniería en Electrónica y Telecomunicaciones], La Libertad: Universidad Estatal Península de Santa Elena, Facultad de Sistemas y telecomunicaciones, 2013.
- [4] C. Montoya Vaque, *Prototipo de robot submarino autónomo tipo torpedo para recopilación de datos oceanográficos* [Tesis de grado de Ingeniería en Electrónica y Telecomunicaciones], La Libertad: Universidad Estatal Península de Santa Elena, Facultad de Sistemas y telecomunicaciones, 2019.
- [5] A. Cadena y S. Vera, «Un vehículo submarino autónomo híbrido y portátil para la exploración antártica,» *Revista de Gestión de Ingeniería y Tecnología*, vol. 07, pp. 39,11, 2020.
- [6] L. Meier, P. Tanskanen, F. Frauendorfer y M. Pollefeys, «PIXHAWK: Un sistema de vuelo autónomo mediante visión artificial a bordo,» *Conferencia internacional IEEE 2011 sobre robótica y automatización*, vol. 33, nº 19, pp. 21-39, 2011.
- [7] R. Durscher, «Auterion,» 01 febrero 2022. [En línea]. Available: <https://auterion.com/pixhawk-powers-the-drone-industry/>.
- [8] Zola y AdiDoks, «Docs.BlueRobotics,» 2020. [En línea]. Available: <https://docs.bluerobotics.com/ardusub-zola/software/onboard/BlueOS-1.0/overview/>.
- [9] k. Martineau, «IBM,» 2022. [En línea]. Available: <https://www.ibm.com/es-es/topics/computer-vision>.
- [10] J. Guillén, «SILO.TIPS,» 2011. [En línea]. Available: <https://silo.tips/download/vision-artificial-aplicada-en-vehiculos-autonomos-submarinos#>.
- [11] B. Noguera, «ingenieriaquimicareviews,» 5 febrero 2021. [En línea]. Available: <https://www.ingenieriaquimicareviews.com/2021/02/flujo-turbulento-en-ingenieria-quimica.html>.
- [12] S. Silva Mendoza, *Diseño e Implementación de robot móvil autónomo de desinfección para navegación social en entornos cerrados y dinámicos*[Tesis de

grado de Ingeniería en Mecatrónica], Escuela Superior Politécnica del Litoral, Facultad de Ingeniería en Mecánica y Ciencias de la Producción, 2021.

- [13] A. Rodriguez, «Nanova,» 5 abril 2023. [En línea]. Available: <https://nanova.org/robots-subacuaticos-que-son-y-como-funcionan/>.
- [14] B. R. Inc, «bluerobotics,» junio 2022. [En línea]. Available: <https://bluerobotics.com/store/>.
- [15] BlueRobotics, «bluerobotics,» 2022. [En línea]. Available: <https://bluerobotics.com/learn/bluerov2-assembly-r3-version/#safety>.
- [16] M. Thinking, «MarineThinking,» 2023. [En línea]. Available: <https://store.marinethinking.com/products/low-light-hd-usb-camera>.
- [17] R. Pi, «raspberrypi,» [En línea]. Available: <https://www.raspberrypi.com/products/camera-module-v2/>.
- [18] ArduSub, «ardusub,» GitBook, [En línea]. Available: <https://www.ardusub.com/introduction/hardware-options/required-hardware.html>.
- [19] J. Bolaños, Implementación de Ardupilot en un UAV tipo Quadrotor para el desarrollo de misiones [Tesis de grado de Ingeniería en Mecatrónica], Universidad Militar Nueva Granada, Facultad de Ingeniería en Mecatrónica, 2014.
- [20] HobbyKing, «hobbyking,» [En línea]. Available: [https://hobbyking.com/en\\_us/turnigy-high-capacity-10000mah-4s-12c-multi-rotor-lipo-pack-w-xt90.html?\\_\\_store=en\\_us](https://hobbyking.com/en_us/turnigy-high-capacity-10000mah-4s-12c-multi-rotor-lipo-pack-w-xt90.html?__store=en_us).
- [21] S. Montoya, «gidahatari,» 21 Julio 2017. [En línea]. Available: <https://gidahatari.com/ih-es/introduccion-al-controlador-de-vuelo-de-drones-pixhawk-hardware-libre>.
- [22] Ardupilot, «ardupilot,» 27 Julio 2023. [En línea]. Available: <https://ardupilot.org/copter/docs/common-pixhawk-overview.html>.
- [23] L. Rodríguez, Implementación de un Sistema de Control de Vuelo Automático para un Vehículo Aéreo no Tripulado [Tesis de de grado de Ingeniería en Electrónica], Instituto Tecnológico de Costa Rica, Escuela de Ingeniería en Electrónica , 2015.
- [24] RaspberryPi, «raspberrypi,» [En línea]. Available: <https://www.raspberrypi.com/products/raspberry-pi-3-model-b/>.
- [25] RaspberryPi, «raspberrypi,» [En línea]. Available: <https://www.raspberrypi.com/products/raspberry-pi-4-model-b/>.
- [26] ArduPilot, «ardupilot,» 11 Febrero 2022. [En línea]. Available: <https://firmware.ardupilot.org/>.



- [27] J. Casado Reina, Desarrollo de un Gateway de MAVLink para Micro Drones para Gestión de vuelo Autónomo [Tesis de grado de Ingeniería], Universidad Carlos III de Madrid, Departamento de Ingeniería Telemática, 2017.
- [28] M. Bouayad, Integración de visión artificial en drones multi-rotor para gestión de misiones autónomas [Tesis de grado en Ingeniería Electrónica Industrial y Automática], Universidad Carlos III de Madrid, 2018.
- [29] B. B. Petisco, Introducción a la Visión Artificial. Procesos y Aplicaciones [Tesis de grado en Ingeniería Matemática], Universidad Complutense de Madrid, Facultad de Ciencias Matemáticas, 2022.
- [30] B. Marr, «Forbes,» 8 abril 2019. [En línea]. Available: <https://www.forbes.com/sites/bernardmarr/2019/04/08/7-amazing-examples-of-computer-and-machine-vision-in-practice/?sh=4cdfb3ae1018>.
- [31] L. Rouhiainen, Inteligencia Artificial: 101 cosas que debes saber hoy sobre nuestro futuro, Primera Edición ed., Barcelona: Alienta Editorial, 2018.
- [32] MATLAB, «mathworks,» [En línea]. Available: <https://la.mathworks.com/discovery/object-detection.html>.
- [33] IBM, «IBM,» 17 agosto 2021. [En línea]. Available: <https://www.ibm.com/docs/es/spss-modeler/saas?topic=networks-neural-model>.
- [34] MATLAB, «mathworks,» [En línea]. Available: <https://la.mathworks.com/discovery/convolutional-neural-network.html>.
- [35] M. Olivares Pérez, Sistema de guiado automático de vehículos mediante técnicas de inteligencia artificial. Aplicación a un PiRacer [Tesis de grado en Ingeniería en Electrónica Industrial y Automática], Universitat Politècnica de València, 2023.
- [36] Z. KEITA, «datacamp,» 2023. [En línea]. Available: <https://www.datacamp.com/es/blog/yolo-object-detection-explained>.
- [37] T. Domínguez Mínguez, Visión artificial: Aplicaciones prácticas con OpenCV - Python, Primera Edición ed., Barcelona: Marcombo, 2021.
- [38] A. T. Parico AIB, «MDPI,» Julio 14 2021. [En línea]. Available: <https://doi.org/10.3390/s21144803>.
- [39] J. Solawetz y S. Sahoo, «Roboflow,» 01 julio 2020. [En línea]. Available: <https://blog.roboflow.com/train-yolov4-tiny-on-custom-data-lighting-fast-detection/>.
- [40] Vieira Douglas, «HostGator,» HostGator, 05 abril 2024. [En línea]. Available: <https://www.hostgator.mx/blog/google-colab/>.

- [41] Intel, «Intel,» 2024. [En línea]. Available: <https://www.intel.la/content/www/xl/es/products/docs/processors/cpu-vs-gpu.html>.
- [42] DataScience, «datascientest,» 21 noviembre 2022. [En línea]. Available: <https://datascientest.com/es/pycharm>.
- [43] Ardusub, «ardusub,» GitBook, [En línea]. Available: <https://www.ardusub.com/quick-start/installing-ardusub.html#completing-calibrations>.

## ANEXOS

### Instalación del sistema operativo

Para poder usar la Raspberry Pi es necesario instalar un sistema operativo, nos dirigimos al sitio oficial de Raspberry Pi <https://www.raspberrypi.com/software/> y descargamos Raspberry Pi Imager (Windows, macOS y Linux), es una herramienta la cual nos permite descargar y escribir imágenes en la Raspberry Pi. Pi Imager viene con algunos sistemas operativos para Raspberry Pi, además permite cargar imágenes de sistemas operativos de proveedores externos para instalar en la Raspberry Pi. Una de las ventajas de Pi Imager es que permite configurar la red Wi-Fi y preconfigurar los ajustes de acceso remoto y poder acceder a la Raspberry Pi.

Una vez instalado Raspberry Pi Imager elegiremos el sistema operativo a utilizar, nos dirigimos al apartado de **CHOOSE OS** como se muestra en la figura 113. Para no tener problemas con la instalación de librerías para la detección de objetos se instalará un sistema operativo de 64 bits, para esto nos vamos a **Raspberry Pi OS (other)** como se muestra en la figura 114 y elegimos el sistema operativo a nuestra conveniencia en este caso es el Raspberry Pi OS 64-bit Debian Bullseye 2023-05-03 como se muestra en la figura 115.

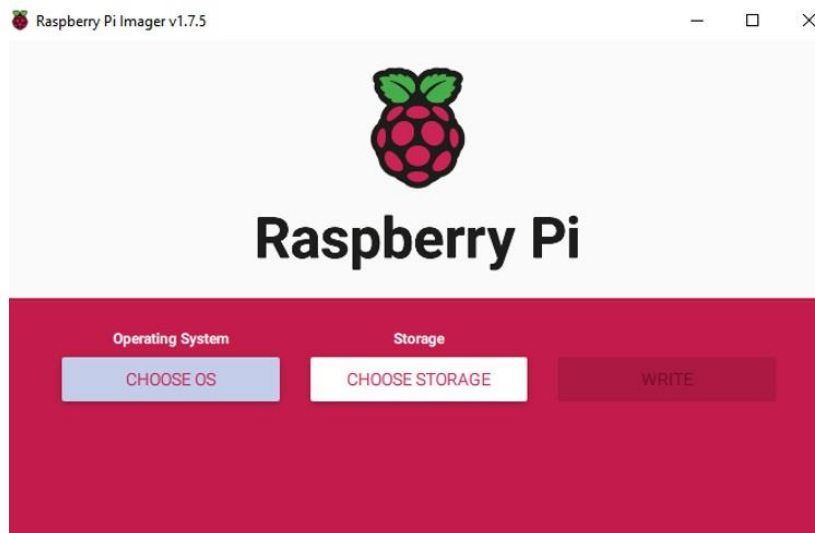


Figura 113: Interfaz de Raspberry Pi Imager

Imagen elaborada por el autor

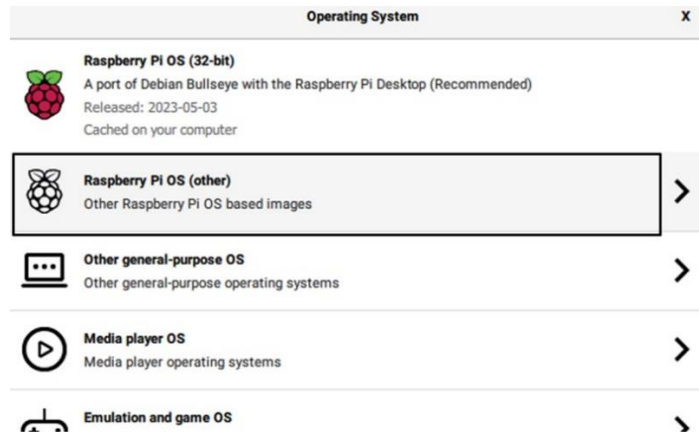


Figura 114: Sistemas Operativos  
Imagen elaborada por el autor

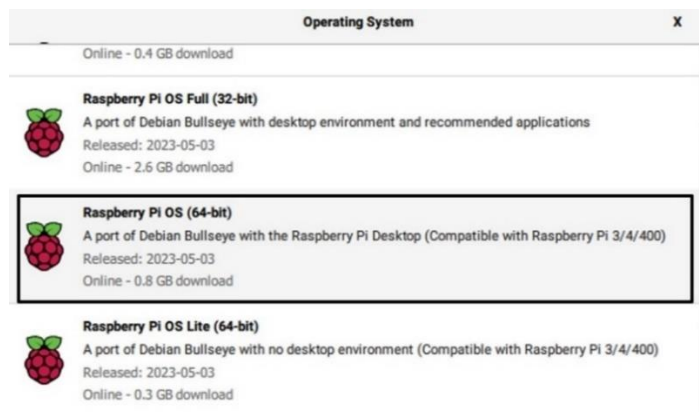


Figura 115: Otras imágenes basadas en el sistema operativo Raspberry Pi  
Imagen elaborada por el autor

Ahora seleccionaremos la tarjeta SD a utilizar para escribir la imagen del sistema operativo, pero antes de eso debemos formatear la tarjeta SD para estar seguros que la tarjeta SD esté libre de archivos basura. Para elegir la tarjeta SD seleccionamos **CHOOSE STORAGE** y elegimos la tarjeta SD que vamos a utilizar

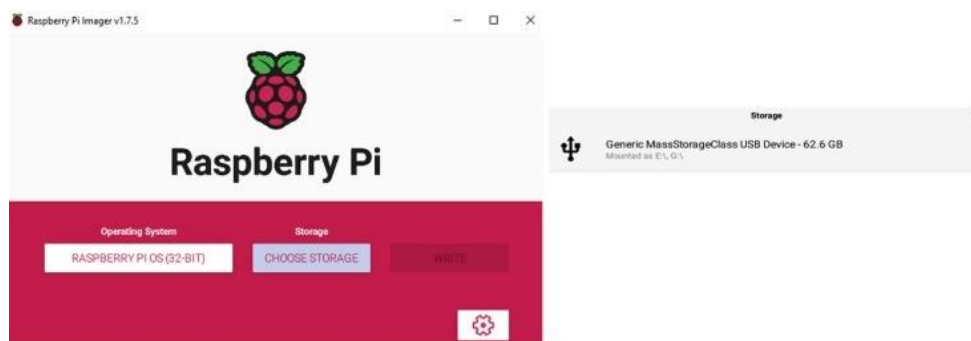


Figura 116: Interfaz Raspberry Pi Imager - Seleccionar tarjeta SD  
Imagen elaborada por el autor

Antes de empezar a escribir la imagen del sistema operativo para la Raspberry Pi seleccionamos la rueda dentada para poder configurar la red Wi-Fi y preconfigurar el acceso remoto a la Raspberry Pi.



Figura 117: Interfaz Raspberry Pi Imager - Seleccionar opciones avanzadas  
Imagen elaborada por el autor

En opciones avanzadas seleccionamos **Enable SSH**, lo cual nos permitirá controlar remotamente la Raspberry Pi, en la parte de **Set username and password** ingresamos el usuario y la contraseña con la que accederemos remotamente a la Raspberry Pi.

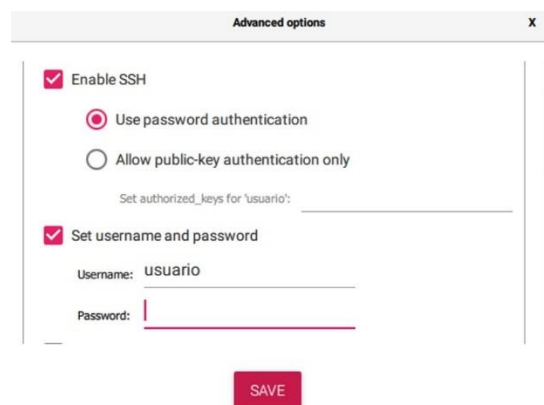


Figura 118: Opciones avanzadas - habilitar SSH  
Imagen elaborada por el autor

Para configurar una red Wi-Fi en la Raspberry Pi habilitamos la opción **Configure Wireless LAN** e introducimos la red y la contraseña a la que nuestra Raspberry se conectara al prenderse, si la red se encuentra oculta habilitamos **Hidden SSID**. Por último, damos clic en **SAVE** para guardar los cambios realizados.

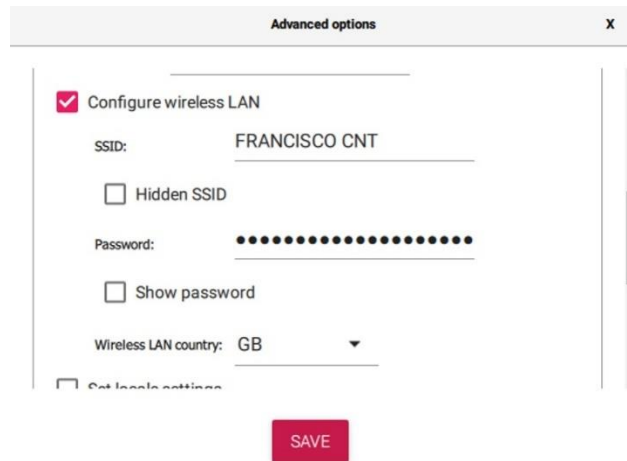


Figura 119: Opciones avanzadas - habilitar red Wi-Fi

Imagen elaborada por el autor

Después de haber seleccionado la imagen de sistema operativo, seleccionar la tarjeta SD, habilitar el SSH y configurar la red Wi-Fi procedemos a seleccionar **WRITE** y se procederá a instalar la imagen de sistema operativo, cuando termine de instalar el sistema operativo nos aparecerá un mensaje que podemos retirar la tarjeta SD.

### **Conexión remota a la Raspberry Pi**

Para poder acceder remotamente a la Raspberry Pi, es necesario el protocolo SSH (Secure Shell), es un protocolo de acceso remoto que le permite a los usuarios controlar y modificar los servidores remotos a los que acceden a través de internet por medio de autenticación para asegurar la transferencia de datos entre el host y el cliente. Como cliente utilizaremos PuTTY desde una computadora portátil para acceder al host que en este caso sería la Raspberry Pi.

### **Identificación de la IP de la Raspberry Pi**

Una vez instalado el sistema operativo a la Raspberry Pi, debemos proceder a encontrar la IP que le fue asignada a la Raspberry Pi. La IP la podemos obtener mediante la conexión Wi-Fi de la Raspberry Pi que se habilitó cuando se instaló el sistema operativo o se conecta un cable de red directamente en la Raspberry Pi.

Para encontrar la IP de la Raspberry Pi se utilizó el programa Advanced IP Scanner, que proporciona una interfaz sencilla de manejar la cual facilitara encontrar la IP asignada ya sea por Wi-Fi o cable de red (ethernet). En la figura 120 se muestra la interfaz de Advanced IP Scanner, damos clic en Explorar y nos mostrara todos los dispositivos que se encuentran conectados a nuestro Router con su respectiva IP. Como se verifico IP por medio de Wi-Fi y cable de red tendremos dos IP asignada para la Raspberry Pi, para la

red Wi-Fi se le asigno la IP 192.168.1.25 y para el cable de red se le asigno la IP 192.168.1.26.

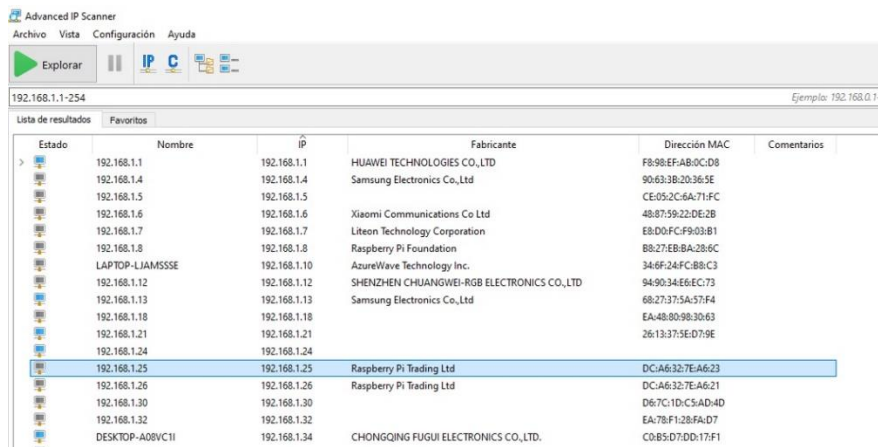


Figura 120: Interfaz de Advanced IP Scanner

Imagen elaborada por el autor

### Acceder a la Raspberry Pi por SSH y habilitar VNC

Al utilizar una computadora portátil con Windows tendremos que utilizar un cliente SSH para poder abrir conexiones SSH, el cliente SSH más utilizado es PuTTY por lo que procederemos a descargar desde su página oficial <https://www.putty.org/> y posteriormente lo instalamos.

Como ya obtuvimos la IP de la Raspberry procedemos a ejecutar el programa PuTTY la cual nos mostrara la interfaz como se muestra en la figura 121, donde tendremos que colocar la IP con la que deseamos acceder por medio la IP dada por Wi-Fi o cable de red.

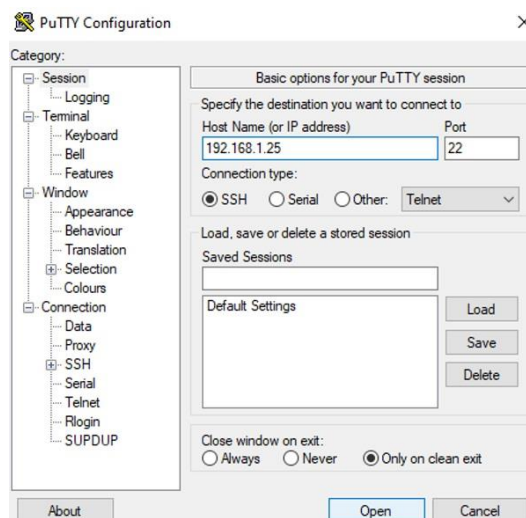


Figura 121: Configuración de PuTTY

Imagen elaborada por el autor

Una vez ingresada la IP nos aparecerá la siguiente interfaz como se muestra en la figura 122, donde tendremos que autenticarnos, como usuario y contraseña se pondrá el nombre y contraseña que se asignó anteriormente.

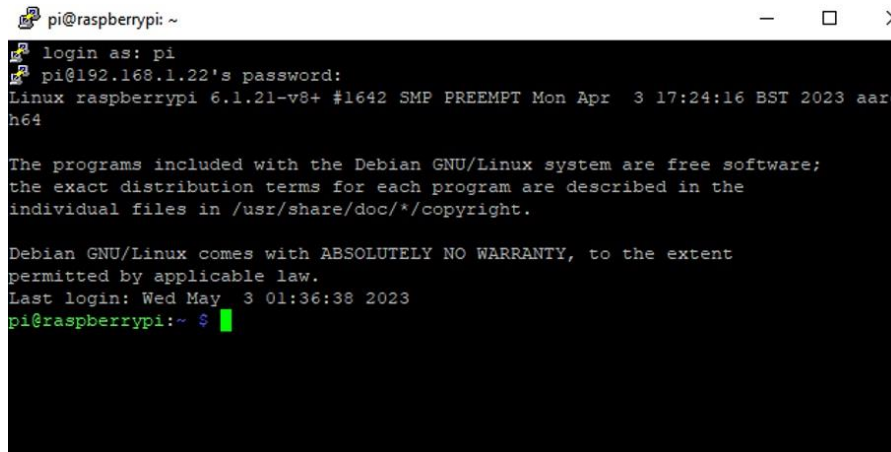


Figura 122: Autenticación para acceso a la Raspberry Pi

Imagen elaborada por el autor

Antes de habilitar VNC tendremos que actualizar el sistema de la Raspberry ingresando los siguientes comandos en el terminal de PuTTY.

*sudo apt-get update*

*sudo apt-get upgrade*

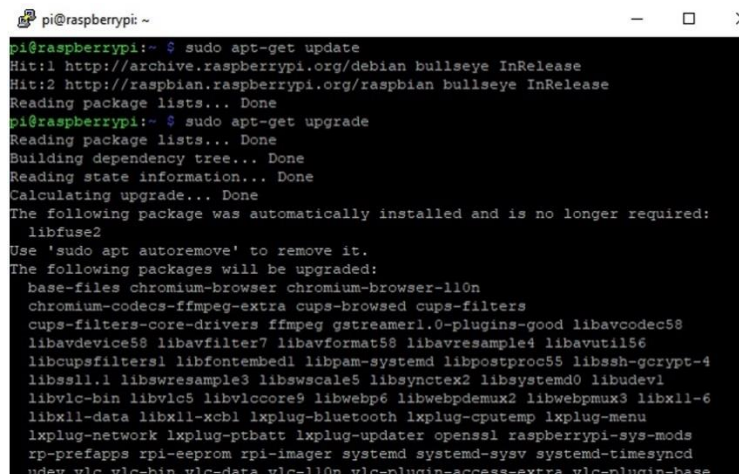


Figura 123: Actualizar el sistema de la Raspberry Pi

Imagen elaborada por el autor



Terminada la actualización del sistema de Raspberry Pi ingresamos el comando `sudo raspi-config` donde nos aparecerá la interfaz como se muestra en la figura 124, donde navegaremos hasta **Interface Options**.

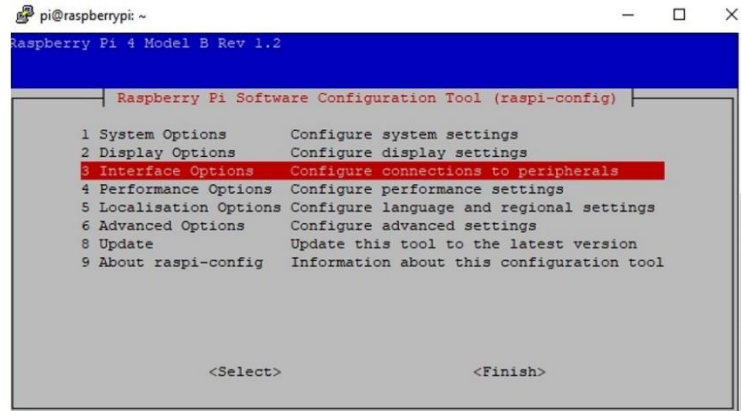


Figura 124: Herramienta de configuración - Interface Options

Imagen elaborada por el autor

En la siguiente venta seleccionamos **VNC**, nos aparecerá otra interfaz donde tendremos que seleccionar **YES** para habilitar VNC en la Raspberry Pi.

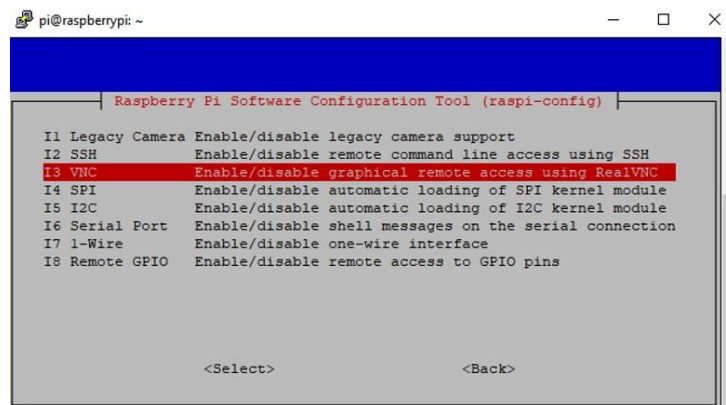


Figura 125: Herramienta de configuración – VNC

Imagen elaborada por el autor

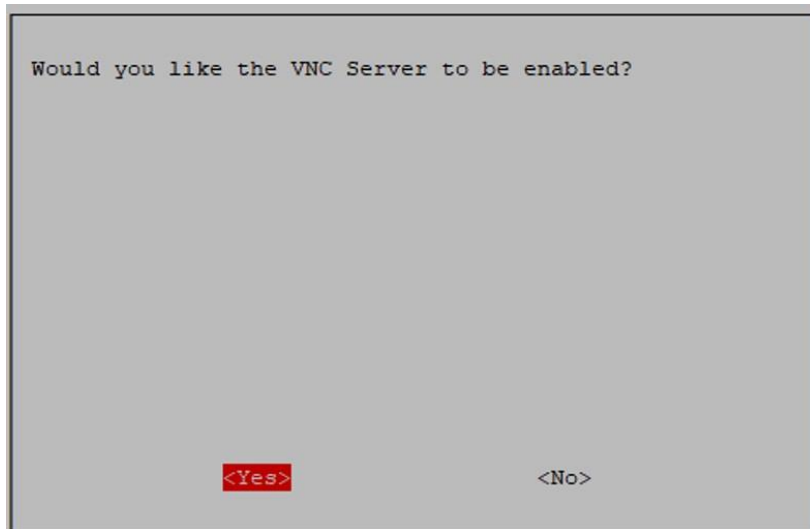


Figura 126: Habilitar VNC  
Imagen elaborada por el autor

Habilitado el VNC configuramos la resolución de la pantalla que tendrá la Raspberry de manera remota, seleccionamos en **Display Options**, luego seleccionamos **VNC Resolution**, seleccionamos **1024x768** y damos en OK, luego de esto nos aparecerá una venta que nos dirá si queremos reiniciar la Raspberry le damos que si para que se guarden los cambios realizados.

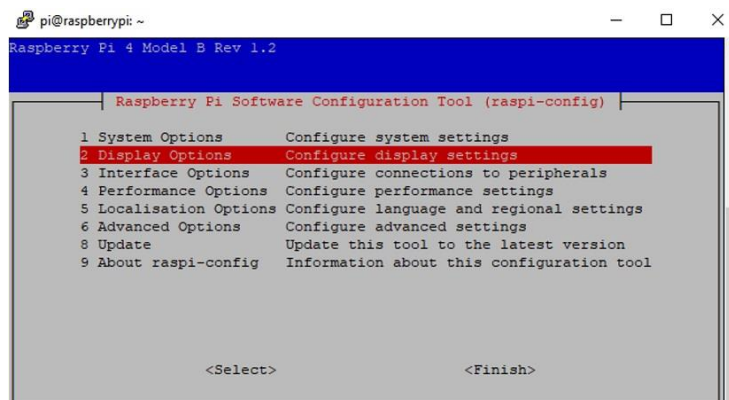


Figura 127:Herramienta de configuración - Display Options  
Imagen elaborada por el autor

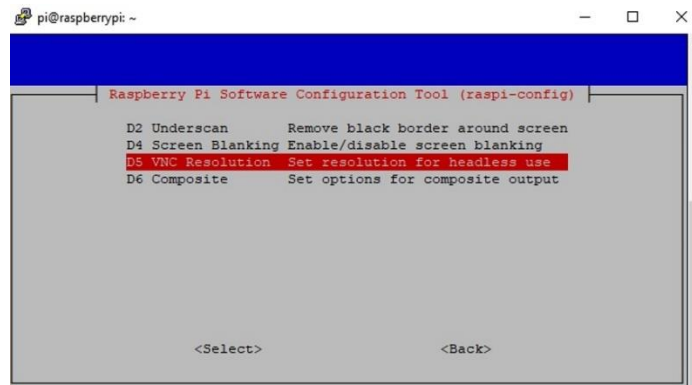


Figura 128: Herramienta de configuración - VNC Resolution

Imagen elaborada por el autor

Habilitado el VNC en la Raspberry Pi tendremos que descargar e instalar RealVNC Viewer en la computadora donde se va a controlar la Raspberry. Instalado RealVNC Viewer lo ejecutamos y creamos una cuenta gratuita para acceder a la interfaz del programa, una vez realizado aquello vamos a la opción de archivo y colocamos nueva conexión, tendremos que ingresar la IP de la Raspberry, damos en aceptar para tener lista la conexión remota a la Raspberry. Como es la primera vez que accederemos a la Raspberry de manera remota nos pedirá usuario y contraseña la cual es la misma que se ingresó cuando se habilito el SSH.

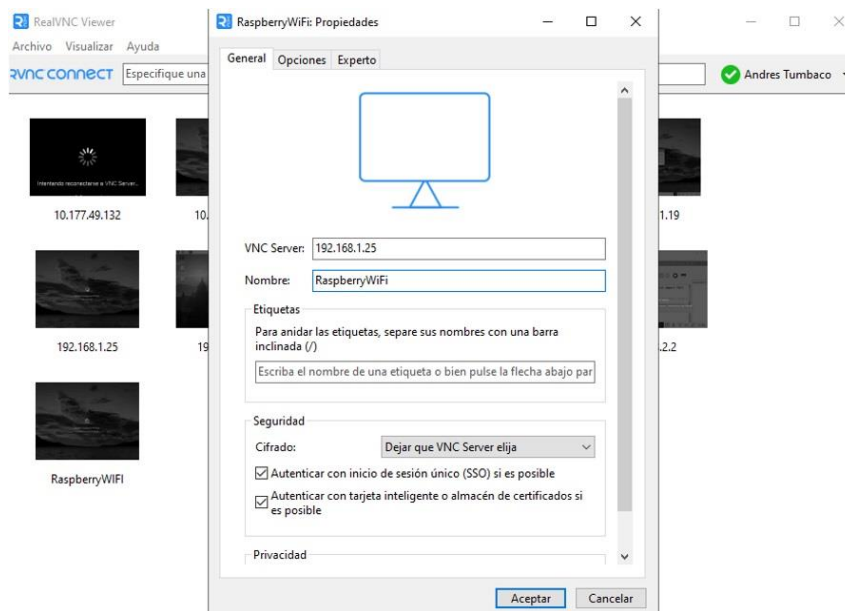


Figura 129: Configuración de IP para VNC Server

Imagen elaborada por el autor

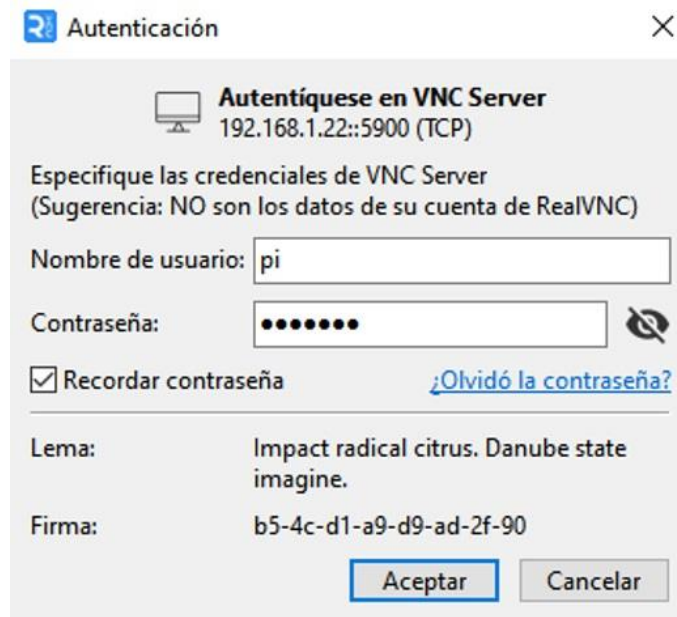


Figura 130: Autenticación en VNC Server

Imagen elaborada por el autor

Ya conectados la Raspberry Pi nos aparecerá esta ventana donde podremos manipular remotamente la Raspberry sin necesidad de un monitor.

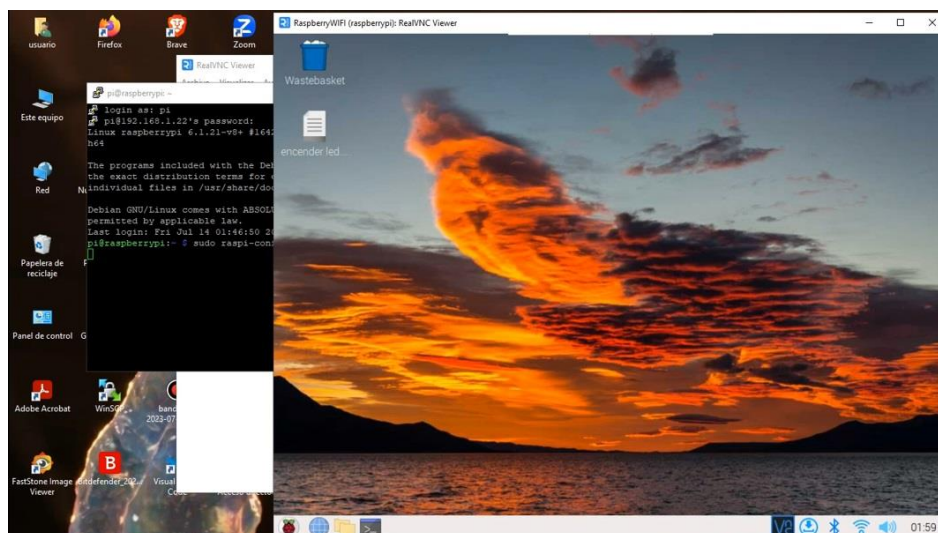


Figura 131: Interfaz gráfica de acceso remoto a la Raspberry Pi

Imagen elaborada por el autor

## Conexión de Robot Submarino a la Estación de Control

### Configuración de Red

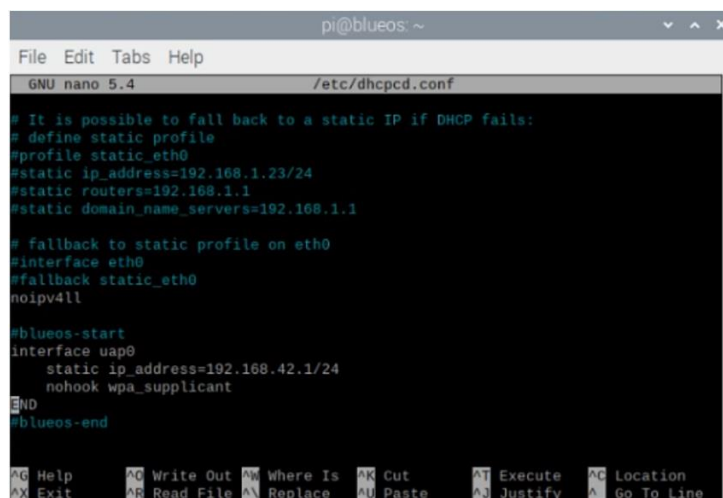
Para entablar la conexión entre el robot submarino y estación de control debemos configurar una IP estática a la Raspberry y la computadora que se utilizara como estación de control.

### Configuración de IP estática a la Raspberry

Accedemos a la Raspberry remotamente por medio de VNC con la IP Wi-Fi que se le asigno anteriormente 192.168.1.25. una vez dentro de la interfaz de la Raspberry abrimos una terminal y digitamos el siguiente comando:

```
sudo nano /etc/dhcpd.conf
```

Como se puede observar en la figura 132 se editará y asignará la IP 192.168.1.1 que deseamos que tenga nuestra Raspberry como ip estática. Realizados los cambios guardamos y reiniciamos la Raspberry.

A screenshot of a terminal window on a Raspberry Pi. The window title is 'pi@blueos: ~'. The terminal shows the nano text editor editing the file '/etc/dhcpd.conf'. The content of the file is as follows:

```
GNU nano 5.4 /etc/dhcpd.conf
# It is possible to fall back to a static IP if DHCP fails:
# define static profile
#profile static_eth0
#static ip_address=192.168.1.23/24
#static routers=192.168.1.1
#static domain_name_servers=192.168.1.1

# fallback to static profile on eth0
interface eth0
#fallback static_eth0
noipv4ll

#blueos-start
interface uap0
  static ip_address=192.168.42.1/24
  nohook wpa_supplicant
END
#blueos-end
```

The bottom of the terminal shows the nano editor's command palette with options like Help, Write Out, Where Is, Cut, Execute, Location, Exit, Read File, Replace, Paste, Justify, and Go To Line.

Figura 132: Configuración de IP Estática a la Raspberry Pi 4

Imagen elaborada por el autor

```
# define static profile
```

```
#profile static_eth0
```

```
static ip_address=192.168.1.1/24
```

```
static routers=192.168.1.1
```

```
static domain_name_servers=192.168.1.1
```

una vez reiniciada la Raspberry verificamos si se guardaron correctamente los cambios, para eso digitamos el siguiente comando *ip addr*.

## Configuración de red de la Estación de control

Abrimos Panel de control y seleccionamos Redes e Internet

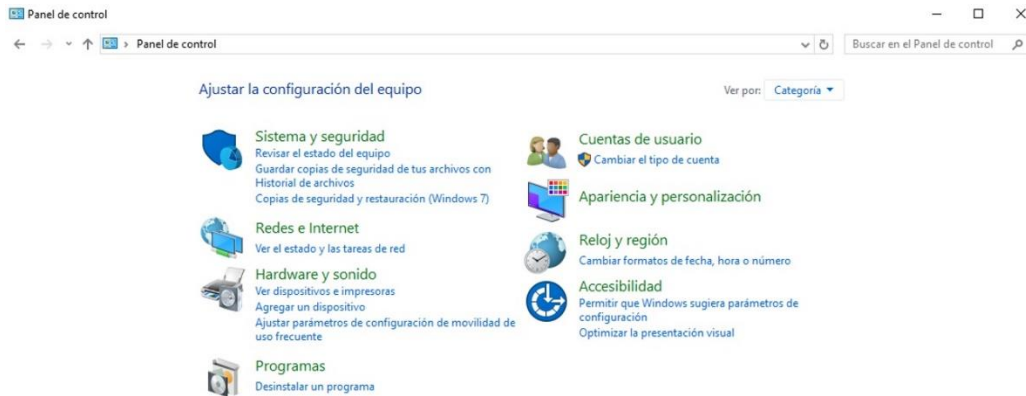


Figura 133: Configuración de IP Estática - Redes e Internet

Imagen elaborada por el autor

En la venta de Redes e Internet seleccionamos Centro de redes y recursos compartidos

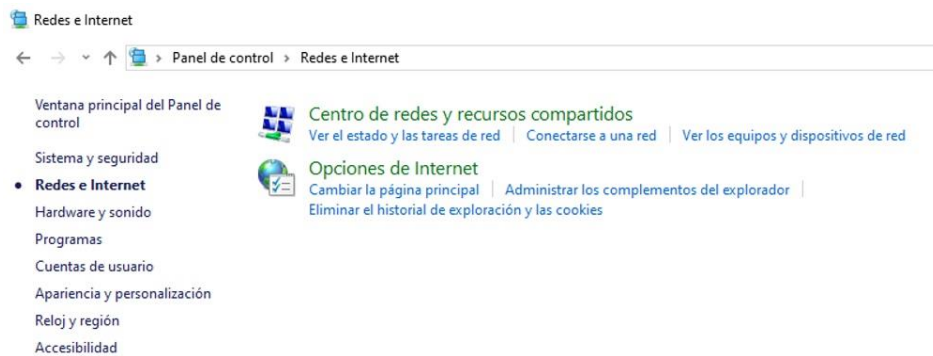


Figura 134: Configuración de IP Estática - Centro de redes y recursos compartidos

Imagen elaborada por el autor

Dentro de Centro de redes y recursos compartidos seleccionamos Cambiar configuración del adaptador.

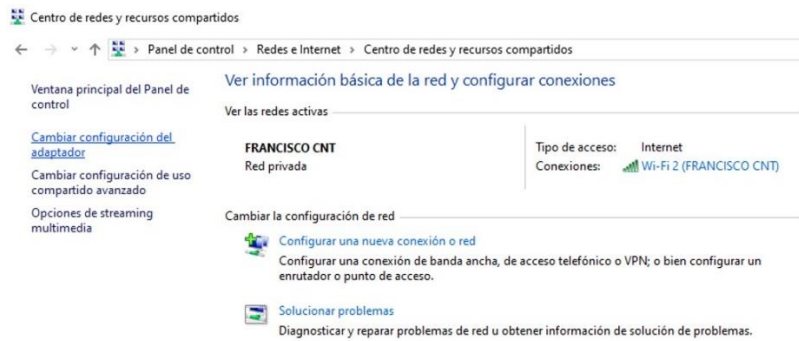


Figura 135: Configuración de IP Estática - Cambiar configuración del adaptador

Imagen elaborada por el autor

Conectamos a la computadora el cable que sale desde la Fathom-X Tether Interface, esta actúa como adaptador USB Ethernet que permitirá que este sea reconocido como un dispositivo Ethernet con el siguiente nombre **Realtek USB FE**. Damos clic derecho sobre este adaptador Ethernet y seleccionamos Propiedades

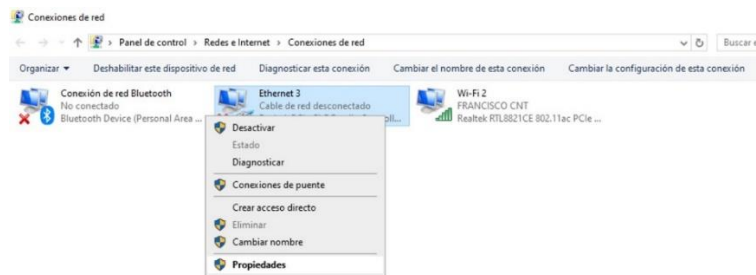


Figura 136: Configuración de IP Estática - USB Ethernet Realtek USB FE

Imagen elaborada por el autor

En la ventana de propiedades del adaptador Ethernet3 seleccionamos Protocolo de Internet versión 4 (TCP/IPv4).

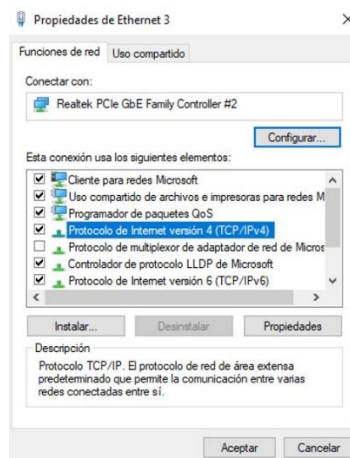


Figura 137: Configuración de IP Estática - Protocolo de internet versión 4 (TCP/IPv4)

Imagen elaborada por el autor

En la siguiente ventana debemos seleccionar Usar la siguiente dirección IP e ingresamos la siguiente ip 192.168.2.1 y para la máscara de subred ingresamos 255.255.255.0 luego damos clic en aceptar para que se guarden los cambios.

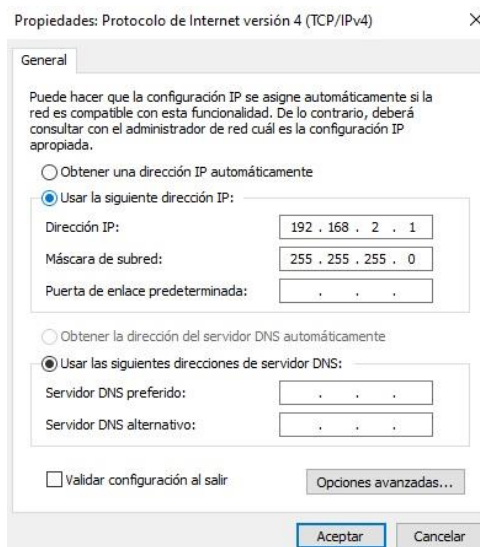


Figura 138: Asignación de IP Estática

Imagen elaborada por el autor

## Personalización del firmware de la placa Pixhawk

El firmware desempeñara un papel importante a la hora de controlar el vehículo ya que este controla los datos adquiridos de los sensores tanto interno y externo que posea el vehículo y así mismo es aquel que genera las salidas hacia los motores del vehículo permitiendo que se pueda controlar en modo manual o autónomo el vehículo. Hay que dejar claro que este paso se podría hacer sin que el Pixhawk se encuentre ubicado en el robot Submarino o bien cuando el robot se encuentre armado pero los pasos serían los mismo.

Para instalar el firmware en el Pixhawk se debe conectar un cable USB-Micro USB desde el Pixhawk a la computadora que tenga instalado el Software Mission Planner o QGround Control.





Figura 139: Pixhawk conectada a PC por medio de cable USB-Micro USB

Imagen elaborada por el autor

Abrimos Mission Planner y seleccionamos el puerto COM que le fue asignado a la placa Pixhawk en la parte superior derecha y también colocamos la tasa de baudios en 56700, pero aún no presionar CONNECT ya que si conectamos la placa Pixhawk al software no nos permitirá instalar el firmware a la placa.



Figura 140: Seleccionamos el Puerto COM6 ArduPilot (COM6)

Imagen elaborada por el autor

Ahora nos dirigimos a **Setup | Install Firmware** nos aparecerá una interfaz donde podremos seleccionar el tipo de vehículo que queremos utilizar. En nuestro caso es un robot submarino debemos seleccionar el firmware Sub.



Figura 141: Instalación de Firmware Sub en Mission Planner

Imagen elaborada por el autor

Seleccionado el tipo de firmware que vamos a instalar en la placa Pixhawk nos aparecerá una ventana donde tendremos que seleccionar el modelo de placa con que se va a trabajar en nuestro caso seleccionamos **Pixhawk 1**, en la parte de Firmwares dejamos por defecto y presionamos en cargar firmware.

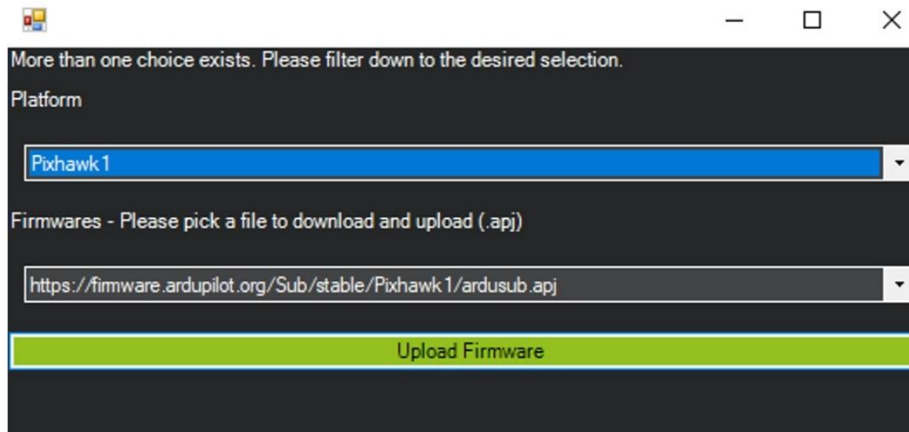


Figura 142: Seleccionar plataforma a utilizar Pixhawk 1

Imagen elaborada por el autor

Cuando seleccionemos el tipo de firmware a utilizar Mission Planner detectara la placa Pixhawk y nos saldrá un mensaje donde nos indica que debemos desconectar la placa presionar OK y volver a conectar, una vez que Mission Planner reconoce la placa empezara la descarga del firmware y su instalación.

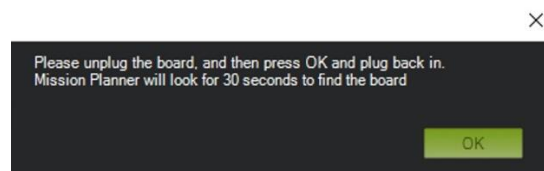


Figura 143: indicaciones a seguir para instalar el firmware Sub en la Pixhawk

Imagen elaborada por el autor

Completado la instalación verificaremos que el firmware se haya instalado correctamente, para esto nos dirigimos a la opción de **Flight Data** y presionamos **CONNECT** en la parte superior derecha del software Mission Planner. Por último, el programa iniciará con la lectura de los parámetros que posee el Firmware Sub, terminado la lectura procedemos a mover la placa Pixhawk y se podrá apreciar en la parte izquierda de Mission Planner responderá a los movimientos que realiza la placa Pixhawk.

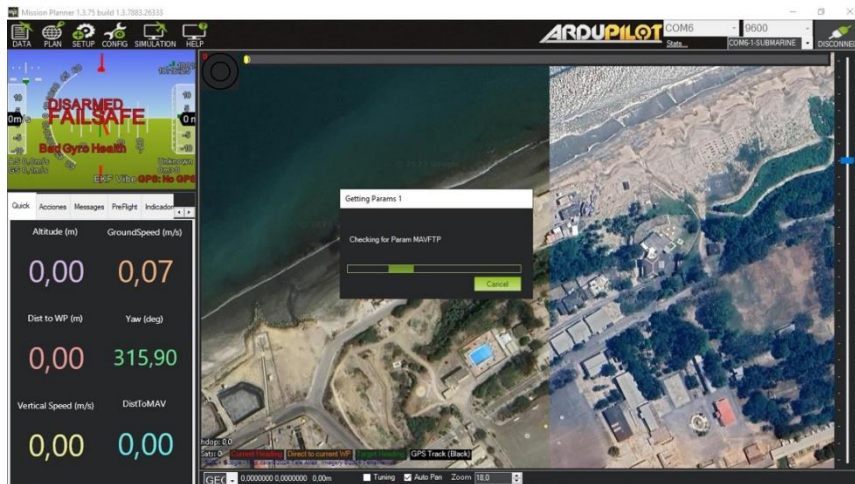


Figura 144: Interfaz de Mission Planner con firmware instalado

Imagen elaborada por el autor

### Configuración del Robot Submarino BlueROV

Para controlar la navegación autónoma del robot submarino se usa el controlador de vuelo Pixhawk. Está diseñado para procesar los datos de los sensores, controla los motores del vehículo y si el robot cuenta servomotores, relés o luces se puede configurar para controlarlos desde el software de la estación terrestre.

La Pixhawk proporciona control manual o autónomo del vehículo, cuenta con varios pines de entrada y salida permitiendo así agregar diferentes periféricos adicionales, además viene incorporado con IMU, giroscopio, acelerómetro y barómetro para brindar estabilidad y dirección al vehículo.

### Configuración de Frame

El robot submarino BlueROV cuenta con 6 propulsores, en la cual debemos elegir un Marco acorde a los motores que posea el robot submarino, para verificar que tengamos el marco correcto se debe ir a la pestaña de **Frame** y elegir **BlueROV2 Vectored**.

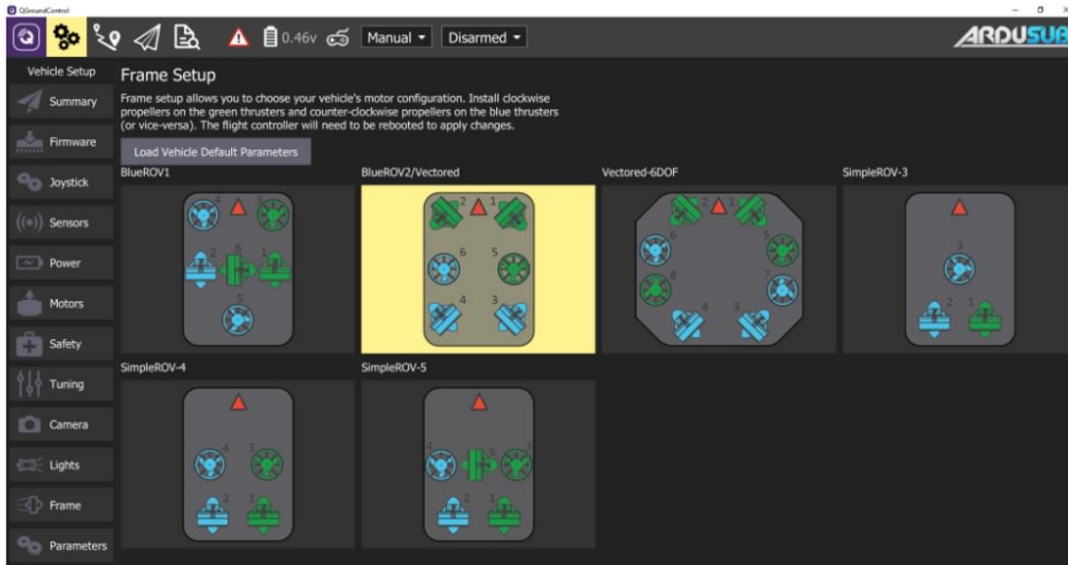


Figura 145: Seleccionamos Frame BlueROV2/Vectored

Imagen elaborada por el autor

Elegido el marco correcto del robot submarino vaya a la pestaña de Motors, donde se verificará que los motores giren en la dirección correcta. Se debe tener en cuenta que los propulsores de color verde giran en sentido contrario a las manecillas del reloj, mientras que los propulsores azules giran en sentido horario.

Verificar el giro de los propulsores es indispensable ya que el sentido depende de cómo fueron conectados en ESC, si se cambió un cable el propulsor girara en sentido contrario provocando que no se pueda controlar de manera el robot. Para activar la prueba de los propulsores se debe deslizar el interruptor.

Mover los controles deslizantes que aparecen en la Figura 146, hará que los propulsores giren donde se podrá verificar si el motor gira de acuerdo al sentido de giro que se mencionó anteriormente. Si el motor no gira en la dirección correcta se debe activar la casilla de verificación del motor que se desea corregir el giro. Hay que tener en cuenta que los propulsores no pueden estar mucho tiempo encendidos si se activan estando fuera del agua ya que estos están lubricados por agua por lo que se podrían dañar si se activan por un tiempo prolongado fuera del agua.

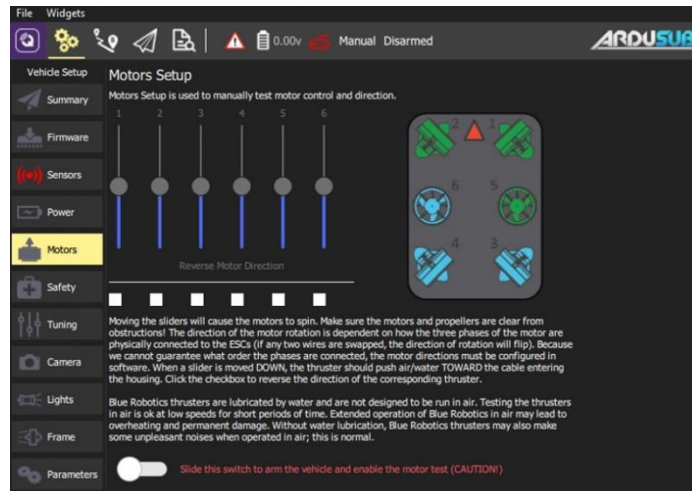


Figura 146: Configuración de giro de los motores  
 Imagen elaborada por el autor

### Calibración de los sensores de la Pixhawk

La calibración del Pixhawk es un parámetro a configurar obligatorio para corregir variaciones fuera del eje. Para realizar la calibración tanto del acelerómetro y del Compass utilizaremos Mission Planner.

### Calibración de Acelerómetro

Antes de empezar la calibración del acelerómetro debemos configurar la orientación que tendrá la placa Pixhawk la cual debe ser Roll90, para cambiar la orientación nos dirigimos a **Configuración | Full Parameter List**, en la lista de parámetros buscamos **AHRS\_ORIENTATION** y colocamos el parámetro **16: Roll90**, para guardar estos cambios presionamos **Write Params**, luego desconectamos la placa Pixhawk y la volvemos a conectar con los cambios ya efectuados.

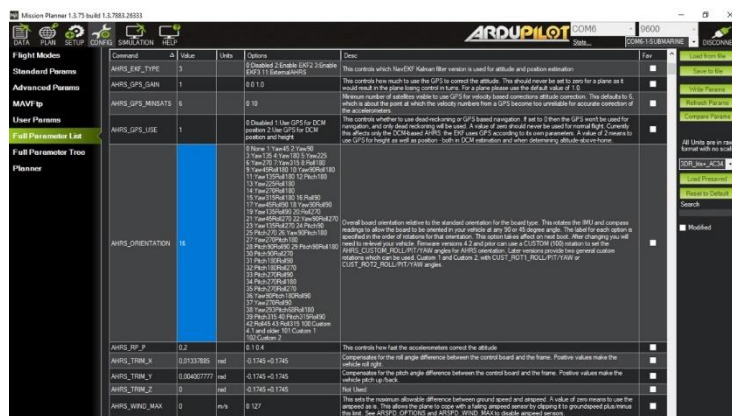


Figura 147: Cambio de orientación en Roll90 para calibrar la Pixhawk  
 Imagen elaborada por el autor

Para calibrar el acelerómetro nos dirigimos a **SETUP | Mandatory Hardware**, luego seleccionamos **Accel Calibration**.

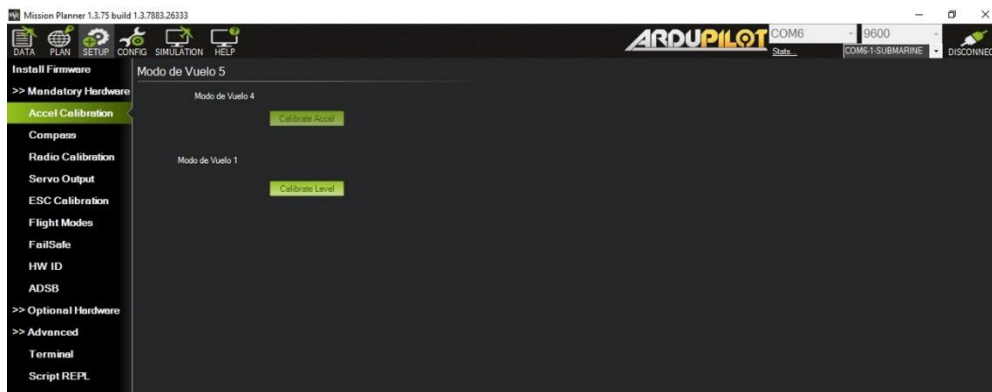


Figura 148: Calibración de acelerómetro de la Pixhawk  
Imagen elaborada por el autor

Presionamos en **Calibrate Accel**, el programa empezara a dar indicaciones de cómo se debe ir colocando la placa Pixhawk en distintas posiciones sin desplazarla.

Level: La primera posición indicada debemos colocar el vehículo nivelado

Left: Colocamos el vehículo de lado izquierdo

Right: Colocamos el vehículo de lado derecho

Nose down: colocar la parte delantera del vehículo apuntando hacia abajo

Nose up: colocar la parte delantera del vehículo apuntando hacia arriba

Back: colocar el vehículo apuntando la parte de abajo hacia arriba

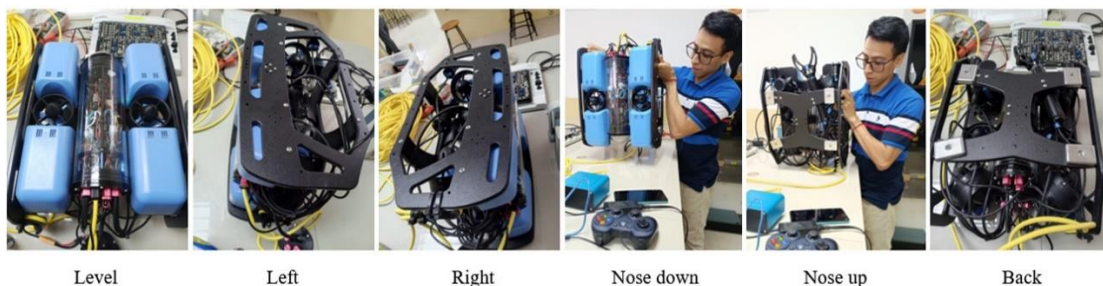


Figura 149: Calibración de acelerómetro – movimiento del robot en las direcciones indicadas  
Imagen elaborada por el autor

Al finalizar la calibración Mission Planner mostrara el mensaje de Calibration Successful.

## Calibración de Compass

Configurar la precisión de la brújula es fundamental para que el vehículo pueda acceder al modo automático ya que este modo depende de información de rumbo, si no se configura de manera correcta el vehículo no se podrá mover a la dirección que se establezca y empezara a dar vueltas sin control. ArduPilot a parte de la brújula interna con la que cuenta la placa Pixhawk permite conectar varias brújulas por medio del puerto I2C. aunque la solo se utiliza la brújula principal de la placa Pixhawk para el modo automático depende de la brújula externa GPS es necesario calibrar estas dos brújulas.

Vamos hacia **Configuración | Mandatory Hardware**, luego seleccionamos **Compass**. Como se van a calibrar la brújula interna y una externa se debe activar la casilla de Compass 1 y Compass 2 y presionar Start, luego se deberá tener el vehículo en el aire y girarlo hacia distintas direcciones de tal modo que todas las caras del vehículo apunten hacia el suelo la calibración terminará cuando se complete por completo las barras verdes de Mag1 y Mag 2.

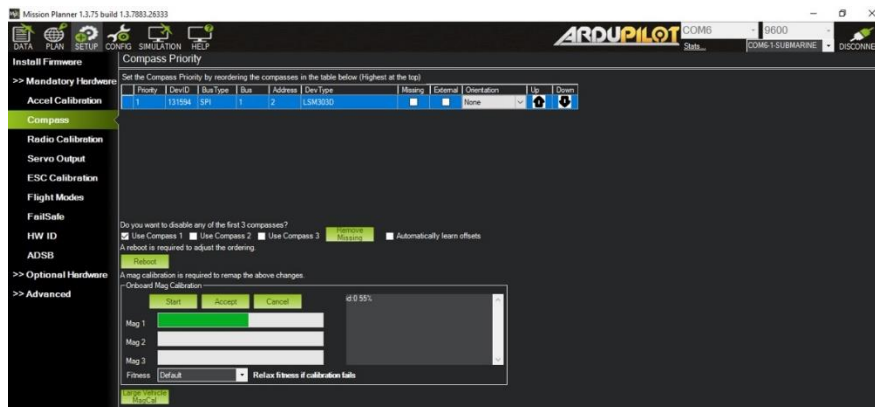


Figura 150: Calibración de compass interno y externo GPS

Imagen elaborada por el autor

Terminada la calibración del Compass aparecerá un mensaje diciendo que debemos reiniciar la placa Pixhawk.

## GPS

La navegación autónoma que realizara el robot submarino se hará por medio de puntos de referencia o Waypoint, por lo que es necesario que el robot cuente con GPS, ya que sin el GPS el robot submarino no podría entrar en modo autónomo debido a que no sabría

la posición en la que se encuentra y no encontraría forma de dirigirse al siguiente punto de la misión autónoma que se planifique.

Para que el robot submarino pueda recibir información del GPS se debe incluir en el EKF (filtro de Kalman Extendido), por lo que se debe configurar los siguientes parámetros:

- En **Mission Planner | CONFIG** seleccionamos **Full Parameter List** y modificamos el parámetro **SERIAL3\_PROTOCOL** ubicamos el valor **2** para configurar como puerto de entrada de comunicación GPS por medio de MAVLink.
- Para que se incluya el GPS en el EKF se debe modificar el parámetro **GPS\_TYPE** donde ubicaremos el valor de **14**

### Personalización de la palanca de mando

La calibración del joystick normalmente no es necesaria ya que el programa por defecto le establece la calibración, en QGround Control en caso que necesite calibración nos indicara con un punto rojo en la pestaña de Joystick. En caso que el joystick requiera de calibración realice lo siguiente.

- Vaya a la pestaña joystick
- Seleccione Calibration y haga clic en Start
- Siga las instrucciones que proporciona QGround Control, donde le indicara en qué dirección mover el joystick izquierdo y derecho de la palanca de mando.

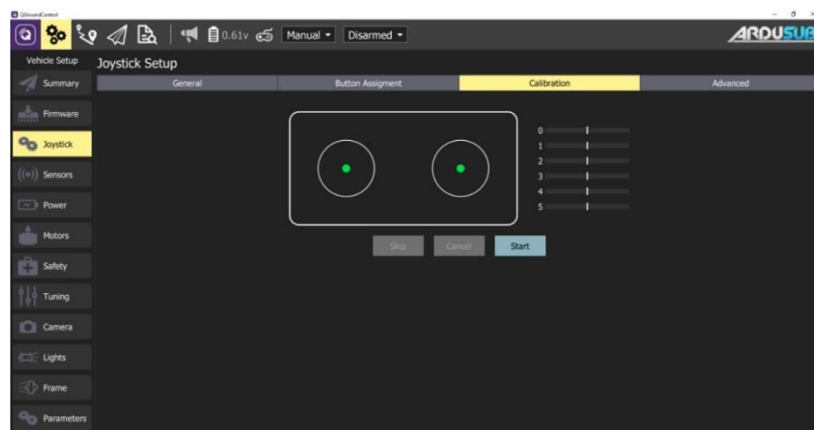


Figura 151: Calibración de joystick

Imagen elaborada por el autor



## Configuración de botones

Es recomendable que configure los botones del joystick de acuerdo a sus preferencias que le permitan maniobrar el robot submarino de manera sencilla, aunque la configuración que proporciona por defecto el programa es sencilla y fácil de manejar. En caso que prefiera reconfigurar los botones del joystick, vaya a la pestaña **Joystick** y seleccione **Button Assignment** donde podrá reconfigurar la configuración a su preferencia.

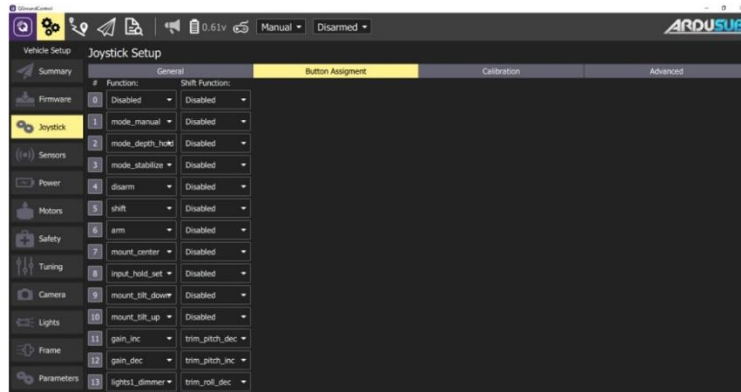


Figura 152: Configuración de botones del joystick

Imagen elaborada por el autor



Figura 153: Configuración predeterminada de botones del joystick [43].

## Configuración de seguridad

En la pestaña de **Safety** podemos activar las alertas de seguridad que tendrá el robot submarino durante la misión autónoma, configurar estos parámetros es indispensable por distintas circunstancias que puedan suceder durante la navegación del submarino. Entre los parámetros importantes a tener en cuenta tenemos:

**Leak (Filtraciones):** Si existen filtración dentro de la cabina de electrónica la estación de control alertara para que empiece el ascenso del robot submarino

**Battery (Batería):** Cando el nivel de voltaje de la batería baje hasta cierto rango el submarino entrara en modo superficie es decir empezara el ascenso.

**Internal Temperature (Temperatura Interna):** Pasado cierta temperatura la estación de control envía alertas contantemente para que el robot ascienda y vuelva a navegar después que la temperatura baje.

**Internal Pressure (Presión Interna):** Si la presión del submarino supera cierto rango establecido la estación de control alertara esta novedad donde se recomienda ascender el robot.

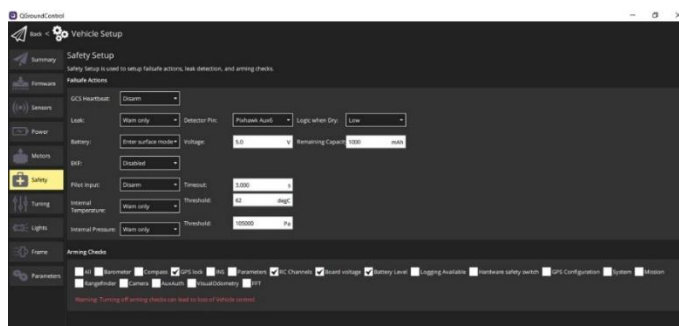


Figura 154: Configuración de seguridad del robot submarino

Imagen elaborada por el autor

## Indicadores de la placa Pixhawk

La Placa Pixhawk presenta algunos estados de parpadeo del Led principal que se debe tener en cuenta ya que nos indican advertencias o si el vehículo se encuentra listo para funcionar, los estados que presenta son los siguientes:

### Parpadeo amarillo

Cuando el led del Pixhawk presenta este parpadeo amarillo nos indica que se ha activado un mecanismo de seguridad, el cual no permitirá el funcionamiento del vehículo. Para saber que mecanismo de seguridad se ha activado, en Mission Planner ir a la pestaña de

mensajes donde mostrara las razones de la activación del mecanismo de seguridad, algunas de las razones pueden ser las siguientes:

- El acelerómetro o brújula requieren calibración.
- Fallo la calibración del acelerómetro o la brújula.
- El acelerador se encuentra por debajo de lo seguro.
- Detección de fuga.

### **Parpadeo azul**

El parpadeo azul en el led de la placa Pixhawk nos indica que el vehículo se encuentra en perfecto estado, pero que no detecta presencia de GPS. Esto significa que el vehículo puede funcionar correctamente en modo manual, pero no funcionara en modos que requieren presencia de GPS como el modo autónomo y guiado lo que imposibilitaría ejecutar misiones autónomas.

### **Parpadeo verde**

La presencia de parpadeo verde en el Led de la placa Pixhawk nos indica que el vehículo no detecta problemas y que se encuentra listo para funcionar.

### **Planificación de misión autónoma**

Teniendo lista la conexión entre la estación de control y el robot submarino, procederemos a planificar la misión autónoma, en esta parte detallaremos como se crean las misiones autónomas en el software Mission Planner y en QGround Control.

Conocer la ubicación del robot submarino es importante antes de programar o planificar una misión autónoma, por lo que es necesario tener una buena recepción de la señal de GPS y que esta se observe en la estación de control.

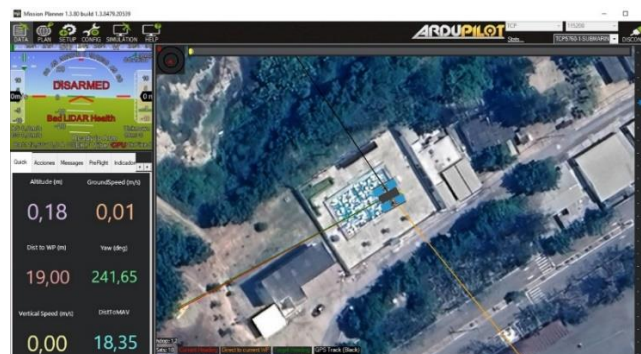


Figura 155: Interfaz de Mission Planner mostrando el robot submarino habilitado por GPS

Imagen elaborada por el autor

## Mission Planner

Para la planificación de misiones autónomas en Mission Planner nos dirigimos a la opción de PLAN. En esta sección realizaremos la planificación de la misión, antes mencionaremos como se encuentra constituida esta sección, en la parte inferior se van registrando los puntos Waypoints que vamos ingresando, en la parte derecha tenemos el panel de acciones donde podremos escribir, leer, guardar y cargar misiones autónomas y por último en la pantalla central tenemos el mapa geográfico que mostrara la ubicación donde se conecta el vehículo.

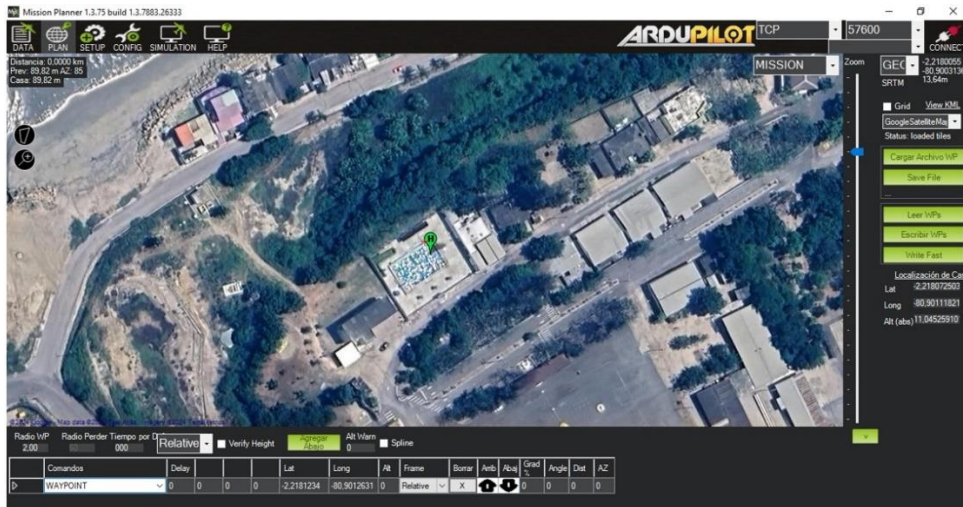


Figura 156: Interfaz en Mission Planner para Planificar Misiones Autónomas

Imagen elaborada por el autor

Hay dos maneras de realizar una misión autónoma, manual o autónoma. La manera autónoma de planificación de una misión es muy utilizada para mapeo de un área y hacer reconocimiento de terreno, este modo de realizar una misión favorece al usuario ya que proporciona varias maneras de generar Waypoints de manera rápida. Para crear la misión autónoma se debe dar clic derecho y elegir la opción dibujar polígono, donde tendrá que marcar un cuadro en el área que desea trabajar.

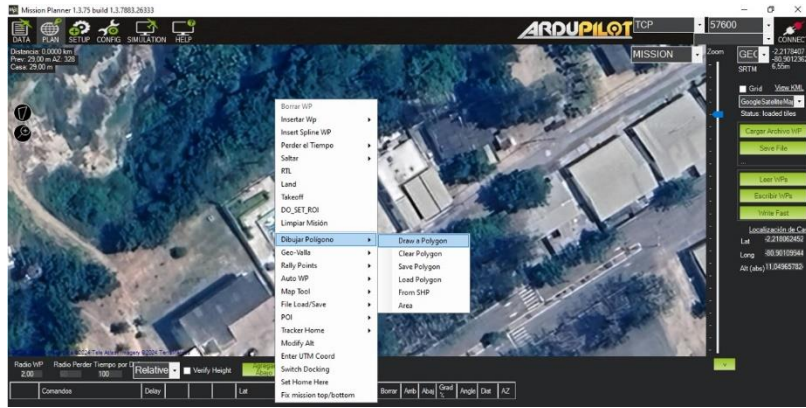


Figura 157: Crear misiones de manera automática - Dibujar Polígono  
Imagen elaborada por el autor

Vuelva a dar clic derecho pero esta vez elija **Auto WP**, en la cual tendrá varias formas de planificar la misión autónoma, en este ejemplo elegimos **SimpleGrid**.

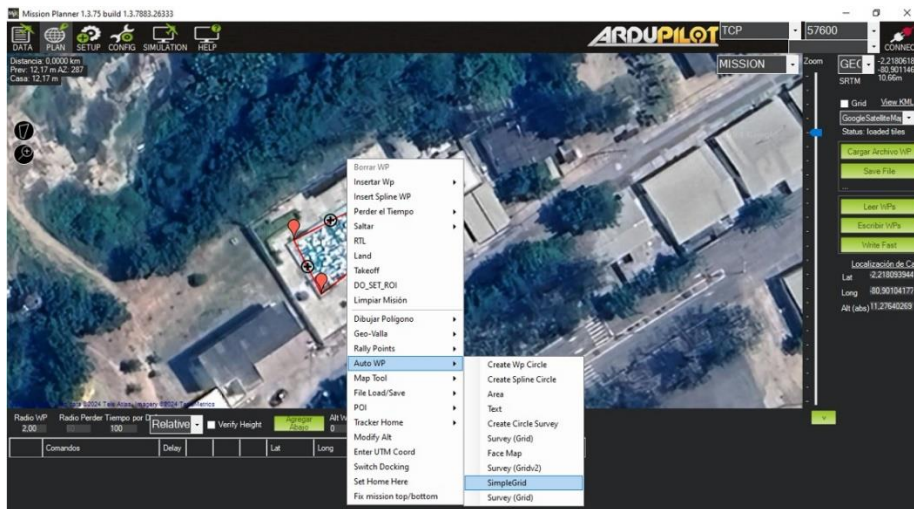


Figura 158: Auto WP – SimpleGrid  
Imagen elaborada por el autor

Aparecerá la siguiente interfaz donde podrá configurar los parámetros de la misión autónoma, haga clic en aceptar y tendrá la misión planificada.

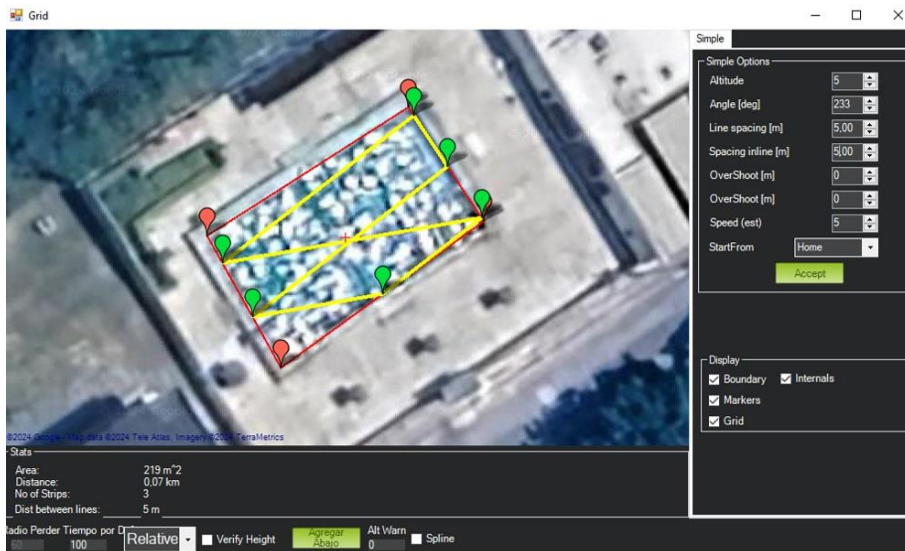


Figura 159: Configuración de Auto WP – SimpleGrid

Imagen elaborada por el autor

En este proyecto la misión autónoma se la planifico de forma manual, para crear esta misión se debe dar clic en el mapa en el lugar que se desea ubicar algún Waypoints, así puede ir agregando hasta completar la misión que desea realizar. En la parte de abajo donde se van registrando los Waypoints se puede modificar los parámetros de **Comandos**, donde podrá modificar las acciones que realice el vehículo, en la parte de **Alt** se debe editar la altitud ya que por defecto la establece como 100, como es un robot submarino se recomienda dejar la altitud en 0 ya Mission Planner no permite ingresar números negativos. Para que la misión autónoma quede grabada en la placa Pixhawk se debe hacer clic en **Escribir WPs** en la parte derecha de la pantalla, donde podrá guardar la misión autónoma y cargarla a la placa Pixhawk en otro momento.



Figura 160: Planificación de misiones autónomas

Imagen elaborada por el autor

La ruta programada para las pruebas que realizará el robot submarino con el software Mission Planner será una ruta sencilla en línea recta como se muestra en la figura 161. Se carga la misión a la placa Pixhawk y nos dirigimos a la opción **DATA**

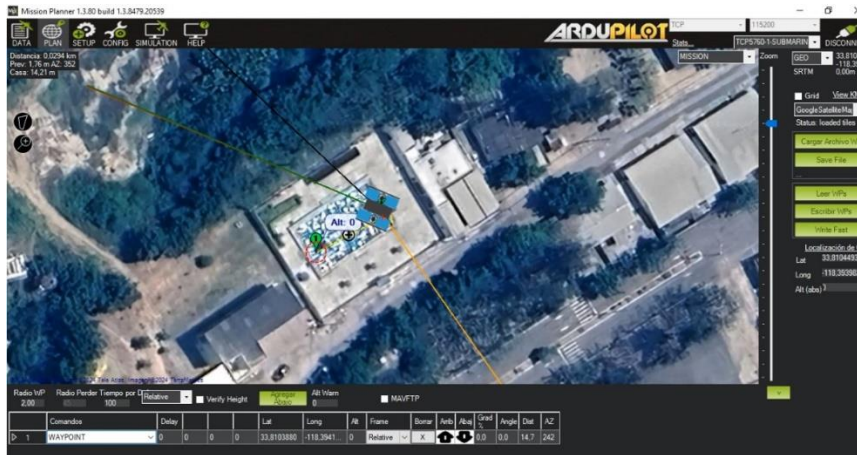


Figura 161: Trayectoria a seguir durante pruebas

Imagen elaborada por el autor

En la opción **DATA** seleccionamos **Acciones**, donde podremos poner en marcha la misión autónoma, tendremos que armar el vehículo para eso damos clic en **Arm/Disarm**, luego en el primer menú desplegable seleccionamos **Mission\_Start** y por último damos clic en **Hacer Acción** y la misión autónoma programada empezara su trayectoria.



Figura 162: Menú de opciones para ejecutar la misión autónoma programada

Imagen elaborada por el autor

## QGround Control

Para crear una misión autónoma en QGround Control vaya a la pestaña **Plan**, luego en la parte izquierda de la pantalla selecciones **File**, se mostrará una interfaz donde podrá elegir crear la misión como **Survey**, **Corridor Scan** y **Structure Scan**. En esa misma interfaz podrá guardar y cargar misiones y limpiar misiones.

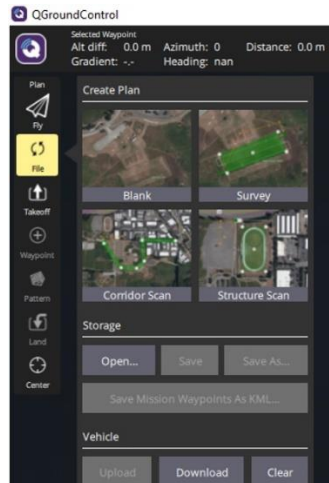


Figura 163: Crear Misiones autónomas en QGroundControl

Imagen elaborada por el autor

Vamos a trabajar con **Structure Scan** para poder planificar la navegación autónoma del robot submarino. Cuando haya seleccionado esta opción le aparecerá un Waypoint de la ubicación del vehículo, para seguir agregando Waypoints dirijase a la parte izquierda de la pantalla y seleccione **Waypoint**, de esta manera podrá seguir agregando Waypoints y planificar la misión autónoma. En la parte derecha de la pantalla se irán registrando los WayPoints que vaya creando donde podrá modificar las acciones que desea que el submarino realice, a cuanta profundidad descenderá el robot, si se mantendrá en la posición durante un tiempo y a qué velocidad realizara la misión autónoma el robot submarino.



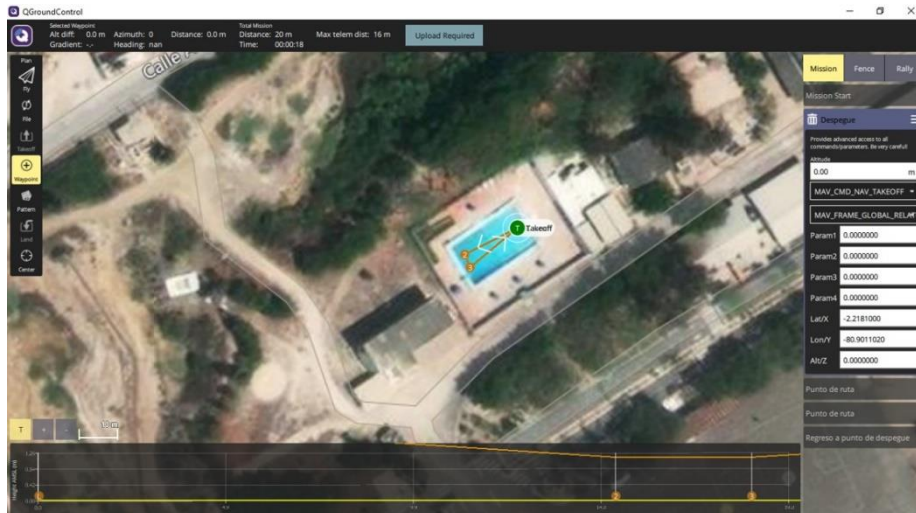


Figura 164: Interfaz de planificación de misión en QGround Control

Imagen elaborada por el autor

En el software QGroundControl se programarán 2 misiones autónomas para realizar pruebas con el robot submarino. La primera será una misión sencilla en línea recta como se muestra en la figura 165 (a) y la segunda será una misión de 4 Waypoints donde el robot submarino ira y regresará formando un rectángulo como se muestra en la figura 165 (b).



(a)

(b)

Figura 165: Misiones autónomas a seguir durante las pruebas en el software QGroundControl

Imagen elaborada por el autor

En la programación de la misión autónoma tendremos configurar algunos parámetros. QGroundControl si permite ubicar valores negativos en el parámetro de altitud para indicar al robot que debe sumergirse en el despegue y seguir la trayectoria hasta llegar al siguiente punto indicado en la ruta. Otro aspecto importante es la velocidad que a la que ira el robot que será de 1.00 m/s el cual es un parámetro recomendado por defecto para iniciar la misión autónoma.

## Configuración de profundidad de descenso al despegue de la misión autónoma



Figura 166: profundidad de despeje de la misión autónoma  
Imagen elaborada por el autor

Ahora tendremos que configurar los parámetros Pitch, Roll y Yaw, BlueRobotics recomienda no modificar demasiado los parámetros ya establecidos ya que podría generar que el robot balancee y cabecee demasiado durante la misión autónoma.

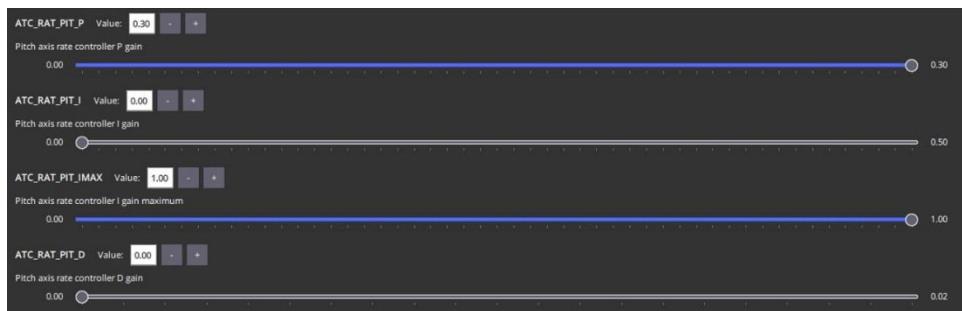


Figura 167: Configuración de parámetro Pitch  
Imagen elaborada por el autor

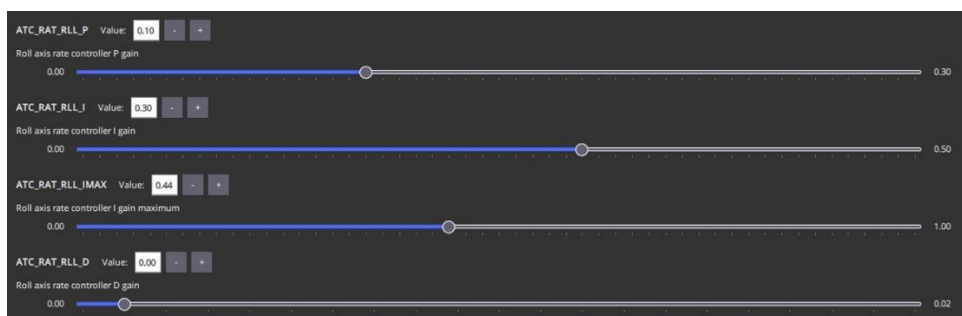


Figura 168: Configuración de parámetro Roll  
Imagen elaborada por el autor



Figura 169: Configuración de parámetro Yaw  
Imagen elaborada por el autor

## Código de navegación autónoma

Se importa librerías a utilizar.

```
import time
from pymavlink import mavutil
```

Permite conectarse al robot submarino a través de MAVLink por medio de IP de la estación de control y puerto UDP.

```
master = mavutil.mavlink_connection('udp:192.168.2.1:14550')
```

Esperar hasta recibir respuesta del robot submarino.

```
master.wait_heartbeat()
```

Función que permite armar al robot.

```
def arm_vehicle():
    print("Armando el vehículo...")
    master.mav.command_long_send(
        master.target_system,
        master.target_component,
        mavutil.mavlink.MAV_CMD_COMPONENT_ARM_DISARM, # Comando para
        armar/desarmar
        0,
```

```

    1, 0, 0, 0, 0, 0, 0 # '1' para armar el vehículo
)
master.motors_armed_wait()
print("Vehículo armado.")

```

función que permite desarmar al robot.

```
def disarm_vehicle():
```

```

    print("Desarmando el vehículo...")
    master.mav.command_long_send(
        master.target_system,
        master.target_component,
        mavutil.mavlink.MAV_CMD_COMPONENT_ARM_DISARM, # Comando para
armar/desarmar
        0,
        0, 0, 0, 0, 0, 0, 0 # '0' para desarmar el vehículo
    )
    master.motors_disarmed_wait()
    print("Vehículo desarmado.")

```

Función que permite enviar comandos de control de movimiento al robot.

```
def send_manual_control(x, y, z, r, duration):
```

```

    target_system = master.target_system
    buttons = 0 # Ningún botón activado
    start_time = time.time()

```

```
while time.time() - start_time < duration:
```

```

    master.mav.manual_control_send(
        target_system,
        x, # Eje X
        y, # Eje Y
        z, # Eje Z (throttle)
        r, # Giro (yaw)
        buttons
    )

```

Comando que permite detener el movimiento de los motores después del tiempo establecido.

```
master.mav.manual_control_send(  
    target_system,  
    0, # Eje X neutral  
    0, # Eje Y neutral  
    500, # Throttle neutro (500 es neutral)  
    0, # Giro neutral  
    buttons  
)
```

Funciones que permiten mover el robot submarino los ejes X, Y, Z, y girar.

```
def move_x(speed, duration):
```

```
# mueve el robot en el eje X en movimientos laterales a la velocidad que se establece  
# durante el tiempo especificado. (el movimiento depende del sentido de giro de los  
# motores que se establezcan en QGroundControl)
```

```
    print(f"Moviendo en el eje X con velocidad {speed} durante {duration} segundos.")  
    send_manual_control(speed, 0, 500, 0, duration)
```

```
def move_y(speed, duration):
```

```
## mueve el robot en el eje X hacia adelante o atrás a la velocidad que se establece  
# durante el tiempo especificado.
```

```
    print(f"Moviendo en el eje Y con velocidad {speed} durante {duration} segundos.")  
    send_manual_control(0, speed, 500, 0, duration)
```

```
def move_z(speed, duration):
```

```
# mueve el robot en el eje Z subir y bajar a la velocidad que se establece durante el  
# tiempo especificado.
```

```
    z_value = 500 + speed # 500 es neutro, se ajusta con la velocidad  
    print(f"Moviendo en el eje Z con velocidad {speed} durante {duration} segundos.")  
    send_manual_control(0, 0, z_value, 0, duration)
```

```
def rotate_yaw(speed, duration):  
# hace girar al robot sobre el eje Z a la velocidad que se establece durante el tiempo  
especificado.
```

```
    print(f"Girando en yaw con velocidad {speed} durante {duration} segundos.")  
    send_manual_control(0, 0, 500, speed, duration)
```

Se llama a las funciones anteriores para armar/desarmar al robot y para mover al robot submarino en la dirección deseada.

```
# Se llama a la función para armar el robot antes de enviar los comandos de  
movimientos  
arm_vehicle()  
# Mueve el robot en el eje Y hacia adelante (Y positivo) a velocidad baja (800) durante  
4 segundos  
move_y(800, 4)  
# mueve el robot en el eje Z, hace descender al robot a la velocidad alta (900) durante 3  
segundos  
move_z(900, 3)  
# hace girar al robot a la velocidad de 900 durante 8 segundos  
rotate_yaw(900, 8)  
# Mueve el robot en el eje Y hacia adelante (Y positivo) a velocidad baja (800) durante  
4 segundos  
move_y(800, 4)  
# desarma al robot submarino después de culminar la misión  
disarm_vehicle()
```

## Encapsular Robot Submarino

Terminada el cambio de componentes y realizado los ajustes necesarios dentro de la capsula electrónica podremos sellarla, para realizar aquello necesitaremos realizar los siguiente:

1. Limpie la tapa de extremo de 18 orificios con toallitas con alcohol isopropílico.
2. Retire las juntas tóricas de la tapa y límpielas
3. Cubra de una capa fina de grasa de silicona las juntas tóricas, no excederse en la aplicación de grasa de silicona.
4. Coloque las juntas tóricas en la tapa de extremo
5. Aplique presión en la tapa de extremo hasta que la capsula quede completamente sellada.



Figura 170: Colocación de grasa de silicona a las juntas tóricas

Imagen elaborada por el autor

## Prueba de vacío del robot submarino

Realizar la prueba de vacío en la capsula electrónica y la de batería es indispensable para asegurarse que estas se encuentren correctamente selladas y evitar que se filtre agua en la capsula electrónica y de batería, los materiales a necesitar son los siguientes:

- Bomba de vacío manual con manómetro
- Bolsa de mangueras de vacío
- Capsula electrónica
- Capsula de batería

Es necesario realizar la prueba de vacío en ambas capsulas al mismo tiempo para asegurarse que ambos compartimientos queden libres de aire ya que ambas capsulan

comparten conexión de aire, para poder realizar aquello la bomba de vacío manual debe contar con dos tapones de vacío que irán conectado en la parte posterior del robot submarino en los tapones de color rojo, asegúrese de colocar de manera correcta la bomba de vacío, luego se deberá bombear al vacío hasta que el manómetro marque aproximadamente unas 15 pulgadas de mercurio en el vacío, una vez llegado a la presión establecida se deberá esperar unos 10 minutos como mínimo para comprobar que el manómetro de la bomba de vacío no pierda presión de vacío durante ese tiempo, si durante este tiempo no se pierde presión nos aseguramos que el robot submarino se encuentra correctamente sellado y el riesgo de filtración de agua es muy bajo.



Figura 171: Bomba de vacío manual con manómetro  
Imagen elaborada por el autor



# Resultado de Antiplagio



## Tesis Andres

**3%**  
Textos sospechosos



**2%** Similitudes  
< 1% similitudes entre comillas  
0% entre las fuentes mencionadas  
**2%** Idiomas no reconocidos

Nombre del documento: Tesis Andres.docx  
ID del documento: acacbf308ae327121271cc59d2a05c037c0eba80  
Tamaño del documento original: 59,75 MB

Depositante: SENDEY AGUSTIN VERA GONZALEZ  
Fecha de depósito: 26/6/2024  
Tipo de carga: Interface  
fecha de fin de análisis: 26/6/2024

Número de palabras: 28.146  
Número de caracteres: 181.330

Ubicación de las similitudes en el documento:



### Fuentes de similitudes

#### Fuentes principales detectadas

| Nº | Descripciones  | Similitudes | Ubicaciones | Datos adicionales                      |
|----|--|-------------|-------------|--|
| 1  | <b>Documento de otro usuario</b> #1w6432<br>El documento proviene de otro grupo<br>1 fuente similar  | < 1%        |             | Palabras idénticas: < 1% (39 palabras) |
| 2  | <b>repositorio.upse.edu.ec</b><br>https://repositorio.upse.edu.ec/bitstream/46000/1389/3/IMPLEMENTACIÓN DE UN PROTOTIPO SU...<br>3 fuentes similares | < 1%        |             | Palabras idénticas: < 1% (77 palabras) |
| 3  | <b>Documento de otro usuario</b> #wffada<br>El documento proviene de otro grupo<br>1 fuente similar  | < 1%        |             | Palabras idénticas: < 1% (33 palabras) |
| 4  | <b>repositorio.upse.edu.ec</b><br>https://repositorio.upse.edu.ec/bitstream/46000/3681/6/UPSE-TN-2016-0040.pdf.bd<br>2 fuentes similares             | < 1%        |             | Palabras idénticas: < 1% (30 palabras) |
| 5  | <b>Documento de otro usuario</b> #1axc92<br>El documento proviene de otro grupo<br>1 fuente similar  | < 1%        |             | Palabras idénticas: < 1% (29 palabras) |

#### Fuentes con similitudes fortuitas

| Nº | Descripciones   | Similitudes | Ubicaciones | Datos adicionales                      |
|----|---|-------------|-------------|--|
| 1  | <b>Documento de otro usuario</b> #763655<br>El documento proviene de otro grupo                                 | < 1%        |             | Palabras idénticas: < 1% (39 palabras) |
| 2  | <b>Documento de otro usuario</b> #f86c168<br>El documento proviene de otro grupo                                | < 1%        |             | Palabras idénticas: < 1% (30 palabras) |
| 3  | <b>Documento de otro usuario</b> #87246c<br>El documento proviene de otro grupo                                 | < 1%        |             | Palabras idénticas: < 1% (16 palabras) |
| 4  | <b>Documento de otro usuario</b> #d98a54<br>El documento proviene de otro grupo                                 | < 1%        |             | Palabras idénticas: < 1% (14 palabras) |
| 5  | <b>repositorio.upse.edu.ec</b><br>https://repositorio.upse.edu.ec/bitstream/46000/4794/1/UPSE-TET-2019-0008.pdf | < 1%        |             | Palabras idénticas: < 1% (22 palabras) |

#### Fuentes mencionadas (sin similitudes detectadas) Estas fuentes han sido citadas en el documento sin encontrar similitudes.

- <https://github.com/freedomwebtech/yolov4tinyrpi4>
- <https://github.com/freedomwebtech/yolov4tinyrpi4.git>
- <https://www.jetbrains.com/es-es/pycharm/>
- <https://github.com/jackonilam/mjpg-streamer/archive/master.tar.gz>
- <http://192.168.2.2:8080/?action=stream>