

## Escuela de Ingeniería Industrial

### TRABAJO FIN DE GRADO

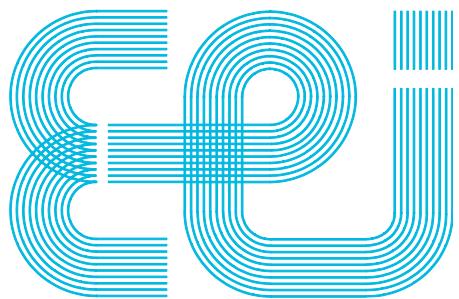
*Clasificación de residuos mediante aprendizaje automático,  
con gestión centralizada de datos a través de comunicaciones de  
largo alcance y bajo consumo*

**Grado en Ingeniería en Electrónica Industrial y Automática**

**ALUMNO:** Daniel Varela Menéndez

**DIRECTORES:** Enrique Paz Domonte

**UniversidadeVigo**



**Escuela de Ingeniería Industrial**

**TRABAJO FIN DE GRADO**

*Clasificación de residuos mediante aprendizaje automático,  
con gestión centralizada de datos a través de comunicaciones de  
largo alcance y bajo consumo*

**Grado en Ingeniería en Electrónica Industrial y Automática**

**Documento**

**MEMORIA**

**UniversidadeVigo**

## ÍNDICE DE CONTENIDO

|        |  |    |
|--------|--|----|
| 1.     | Introducción .....                                     | 12 |
| 1.1.   | Planteamiento del problema .....                       | 12 |
| 1.2.   | Precedentes y otras alternativas .....                 | 13 |
| 2.     | Solución propuesta y justificación .....               | 14 |
| 3.     | Resumen y Objetivos .....                              | 15 |
| 4.     | Implementación de una red neuronal .....               | 16 |
| 4.1.   | Principios básicos. Perceptrón .....                   | 16 |
| 4.2.   | Principios básicos. Redes neuronales .....             | 18 |
| 4.2.1  | Función de coste .....                                 | 18 |
| 4.2.2. | Descenso del gradiente y optimizadores .....           | 20 |
| 5.     | Red neuronal .....                                     | 29 |
| 5.1.   | Set de imágenes .....                                  | 29 |
| 5.2.   | Reentrenamiento de la red .....                        | 31 |
| 5.2.1. | Entrenamiento y validación.....                        | 31 |
| 5.2.2. | Preprocesamiento de las imágenes .....                 | 31 |
| 5.2.3. | Construcción de la red.....                            | 32 |
| 5.2.4. | Compilación y entrenamiento .....                      | 36 |
| 5.3.   | Test y resultados obtenidos .....                      | 40 |
| 5.4.   | Congelación del modelo e inferencia .....              | 41 |
| 6.     | Diseño del prototipo.....                              | 43 |
| 6.1.   | Mecanismo compuertas .....                             | 44 |
| 6.1.1. | Sujeción compuerta .....                               | 49 |
| 6.1.2. | Compuerta (manivela).....                              | 50 |
| 6.1.3. | Eslabón (biela).....                                   | 52 |
| 6.1.4. | Émbolo .....   | 53 |
| 6.1.5. | Sujeción Correa .....                                  | 53 |
| 6.1.6. | Sujeción varillas .....                                | 54 |
| 6.1.7. | Porta poleas .....                                     | 55 |
| 6.1.8. | Sujeción motor compuertas .....                        | 55 |
| 6.2.   | Mecanismo selector .....                               | 57 |
| 6.2.1. | Conector tubo con la parte superior de la máquina..... | 58 |
| 6.2.2. | Porta-ruedas.....                                      | 59 |
| 6.2.3. | Porta-ruedas auxiliar .....                            | 59 |
| 6.2.4. | Engranaje conducido .....                              | 60 |
| 6.2.5. | Engranaje conductor.....                               | 60 |

|         |   |     |
|---------|---|-----|
| 6.2.6.  | Sujeción del rodamiento.....                                | 61  |
| 6.2.7.  | Sujeción del motor del selector .....                       | 62  |
| 6.2.8.  | Sujeción final de carrera del selector .....                | 63  |
| 6.3.    | Estructura para el asentamiento de la máquina .....         | 64  |
| 6.3.1.  | Fijación tubo horizontal .....                              | 64  |
| 6.3.2.  | Codo 90° .....  | 65  |
| 6.3.3.  | Pie.....  | 66  |
| 6.4.    | Otras piezas.....   | 66  |
| 6.4.1.  | Sujeción ESP32 (LoRa) .....                                 | 66  |
| 6.4.2.  | Sujeción antena LoRa.....                                   | 67  |
| 6.4.3.  | Sujeción cámara .....                                       | 68  |
| 6.4.4.  | Sujeción sensor de ultrasonidos .....                       | 70  |
| 6.4.5.  | Plantilla para serigrafía.....                              | 71  |
| 7.      | Impresión 3D y construcción del prototipo .....             | 72  |
| 7.1.    | Impresión 3D .....  | 72  |
| 7.1.1.  | Parámetros de impresión .....                               | 72  |
| 7.2.    | Construcción.....   | 74  |
| 8.      | Componentes electrónicos. estructura de conexiones.....     | 80  |
| 8.1.    | Jetson Nano .....   | 80  |
| 8.2.    | Arduino Mega.....   | 81  |
| 8.2.1.  | Movimiento de compuertas .....                              | 82  |
| 8.2.2.  | Movimiento del selector.....                                | 82  |
| 8.2.3.  | Señalización del residuo a procesar .....                   | 83  |
| 8.3.    | ESP32 LoRa .....  | 83  |
| 9.      | Programación .....  | 85  |
| 9.1.    | Jetson Nano .....   | 85  |
| 9.2.    | Arduino Mega.....   | 86  |
| 10.     | Comunicación de los datos mediante LoRa .....               | 92  |
| 10.1.   | Emisor LoRa.....  | 93  |
| 10.1.1. | Importación de librerías .....                              | 94  |
| 10.1.2. | Definición de variables.....                                | 94  |
| 10.1.3. | Función de inicialización de la OLED .....                  | 96  |
| 10.1.4. | Función de inicialización del LoRa.....                     | 96  |
| 10.1.5. | Función de medición del nivel .....                         | 97  |
| 10.1.6. | Función de actualización de variables.....                  | 97  |
| 10.1.7. | Función para emitir el paquete de datos mediante LoRa ..... | 98  |
| 10.1.8. | Función Setup.....  | 99  |
| 10.1.9. | Función Loop .....  | 100 |
| 10.2.   | Receptor LoRa.....  | 101 |
| 10.2.1. | Importación de librerías .....                              | 102 |

|          |  |     |
|----------|--|-----|
| 10.2.2.  | Definición de variables.....                                 | 103 |
| 10.2.3.  | Función inserta valores en la web .....                      | 105 |
| 10.2.4.  | Función de inicialización de la OLED .....                   | 105 |
| 10.2.5.  | Función de inicialización del LoRa.....                      | 106 |
| 10.2.6.  | Función de conexión WiFi .....                               | 107 |
| 10.2.7.  | Función de recepción del paquete y extracción de datos ..... | 108 |
| 10.2.8.  | Función de obtención de la fecha y hora.....                 | 108 |
| 10.2.9.  | Función Setup.....   | 110 |
| 10.2.10. | Función Loop.....  | 111 |
| 10.3.    | Diseño del servidor web .....                                | 111 |
| 10.3.1.  | Cabecera .....   | 113 |
| 10.3.2.  | Cuerpo .....   | 114 |
| 11.      | Desarrollo aplicación móvil .....                            | 116 |
| 12.      | Breve estudio económico .....                                | 118 |
| 13.      | Conclusión y posibles mejoras.....                           | 120 |
| 14.      | Bibliografía.....  | 122 |
| 15.      | Anexos.....  | 125 |
| 15.1.    | Planos .....   | 126 |
| 15.2     | Códigos de programas .....                                   | 127 |
| 15.2.1   | Código de la red neuronal .....                              | 128 |
| 15.2.2   | Código del script para generar un dataset .....              | 129 |
| 15.2.3   | Código del Jetson Nano.....                                  | 130 |
| 15.2.4   | Código de Arduino Mega.....                                  | 131 |
| 15.2.5   | Código del emisor LoRa .....                                 | 132 |
| 15.2.6   | Código del receptor LoRa .....                               | 133 |
| 15.2.7   | Código del servidor web .....                                | 134 |
| 15.2.8   | Código de la aplicación móvil.....                           | 135 |

## ÍNDICE DE ILUSTRACIONES

|  |    |
|--|----|
| Figura 1: Niveles de reciclaje en países de la Unión Europea .....   | 12 |
| Figura 2: Tambores separadores magnéticos .....  | 13 |
| Figura 3: Diagrama inteligencia artificial, aprendizaje automático y profundo .....  | 16 |
| Figura 4: Estructura de un perceptrón.....   | 17 |
| Figura 5: Función de entropía cruzada categórica .....   | 18 |
| Figura 6: Gráfica de la pérdida de una función de entropía cruzada respecto a que la probabilidad predecida sea la correcta..... | 19 |
| Figura 7: Interpretación del descenso del gradiente .....  | 20 |
| Figura 8: Punto de silla de caballo .....  | 22 |
| Figura 9: Curva patológica.....  | 24 |
| Figura 10: Trayectoria de descenso sobre una curva patológica .....  | 24 |
| Figura 11: Componentes del gradiente sobre las trayectorias de zigzag en una curva patológica .....                              | 26 |
| Figura 12: Proceso de creación del dataset .....   | 30 |
| Figura 13: Precisión red frente a operaciones de las principales redes dedicadas a clasificación .....                           | 32 |
| Figura 14: Arquitectura Inception V3 .....   | 33 |
| Figura 15: Resultados de error de diferentes redes en el certamen ILSVRC de 2015.....  | 34 |
| Figura 16: Ejemplo de reentrenamiento de las últimas capas de Inception-v3.....  | 34 |
| Figura 17: Arquitectura de la red construida .....   | 36 |
| Figura 18: Precisión de entrenamiento frente a épocas de los optimizadores probados superpuestos .....                           | 37 |
| Figura 19: Precisión de validación frente a épocas de los optimizadores probados superpuestos .....                              | 37 |
| Figura 20: Pérdida durante el entrenamiento de los optimizadores probados .....  | 38 |
| Figura 21: Pérdida de validación de los optimizadores probados.....  | 38 |
| Figura 22: Máximos de precisión de los optimizadores empleados .....   | 39 |
| Figura 23: Precisión del SGD con una tasa de aprendizaje del 0,001 .....   | 39 |
| Figura 24: Pérdida del SGD con una tasa de aprendizaje del 0,001 .....   | 40 |
| Figura 25: Máximos de precisión del SGD con una tasa de aprendizaje del 0,001 .....  | 40 |
| Figura 26: Resultados del test sobre la red elegida.....   | 41 |
| Figura 27: Vista general del modelo 3D de la máquina.....  | 43 |
| Figura 28: Vista general del interior de la máquina.....   | 43 |
| Figura 29: Vista general del mecanismo de compuertas .....   | 44 |
| Figura 30: Mecanismo de compuertas. Compuertas cerradas.....   | 45 |
| Figura 31: Mecanismo de compuertas. Cálculos geométricos de diseño de compuertas cerradas .....                                  | 46 |
| Figura 32: Mecanismo de compuertas. Compuertas abiertas .....  | 47 |
| Figura 33: Mecanismo de compuertas. Cálculos geométricos de diseño con compuertas abiertas .....                                 | 48 |
| Figura 34: Sujeción de compuerta .....   | 49 |

---

|   |    |
|---|----|
| Figura 35: Compuerta. Vista superior delantera .....                                      | 50 |
| Figura 36: Compuerta. Vista superior trasera .....  | 51 |
| Figura 37: Compuerta. Esquema de referencia para distancias .....                         | 51 |
| Figura 38: Eslabón .....  | 52 |
| Figura 39: Émbolo .....   | 53 |
| Figura 40: Sujeción correa .....  | 53 |
| Figura 41: Sujeción varillas .....  | 54 |
| Figura 42: Porta poleas .....   | 55 |
| Figura 43: Sujeción del motor de la compuerta. Motor 28BYJ-48 insertado .....             | 55 |
| Figura 44: Sujeción del motor de la compuerta. Vista delantera .....                      | 56 |
| Figura 45: Sujeción del motor de la compuerta. Vista trasera .....                        | 56 |
| Figura 46: Mecanismo del selector en el contexto de la máquina .....                      | 57 |
| Figura 47: Mecanismo del selector. Vista general .....                                    | 57 |
| Figura 48: Conector tubo con la parte superior. Vista superior .....                      | 58 |
| Figura 49: Conector tubo con la parte superior. Vista inferior .....                      | 58 |
| Figura 50: Porta-ruedas .....   | 59 |
| Figura 51: Porta-ruedas auxiliar .....  | 59 |
| Figura 52: Engranaje conducido .....  | 60 |
| Figura 53: Engranaje conductor. Vista general .....                                       | 60 |
| Figura 54: Engranaje conductor. Vista de la hendidura para insertar el rotor .....        | 61 |
| Figura 55: Sujeción del rodamiento .....  | 61 |
| Figura 56: Sujeción del motor del selector NEMA 17 insertado .....                        | 62 |
| Figura 57: Sujeción del motor del selector. Vista de la zona de inserción del motor ..... | 62 |
| Figura 58: Sujeción del final de carrera del selector, en el contexto del mecanismo ..... | 63 |
| Figura 59: Sujeción del final de carrera del selector .....                               | 63 |
| Figura 60: Estructura para el asentamiento de la máquina .....                            | 64 |
| Figura 61: Fijación tubo horizontal .....   | 64 |
| Figura 62: Codo 90°. Vista lateral inferior .....   | 65 |
| Figura 63: Codo 90°. Vista lateral superior .....   | 65 |
| Figura 64: Pie .....  | 66 |
| Figura 65: Sujeción ESP32. ESP32 insertado .....  | 66 |
| Figura 66: Sujeción ESP32 .....   | 67 |
| Figura 67: Sujeción antena LoRa. Vista en el contexto de la máquina .....                 | 67 |
| Figura 68: Sujeción antena LoRa .....   | 68 |
| Figura 69: Sujeción cámara. Colocación y ángulo .....                                     | 68 |
| Figura 70: Sujeción de la cámara. Colocación vista de perfil .....                        | 69 |
| Figura 71: Sujeción de la cámara .....  | 69 |
| Figura 72: Sujeción del sensor de ultrasonidos, en el contexto de la máquina .....        | 70 |
| Figura 73: Sujeción del sensor de ultrasonidos .....                                      | 70 |
| Figura 74: Plantilla para serigrafía .....  | 71 |
| Figura 75: Vista general de la maquina ensamblada .....                                   | 74 |

---

|   |     |
|---|-----|
| Figura 76: Pared de la maquina pintada.....   | 75  |
| Figura 77: Proceso de pintado de la serigrafía.....   | 75  |
| Figura 78: Compuerta y sujeción pintadas.....   | 76  |
| Figura 79: Sujeción, compuerta, eslabón, émbolo y sujeción de la correa ensamblados ..        | 76  |
| Figura 80: Vista exterior del mecanismo de la compuerta ensamblado .....                      | 77  |
| Figura 81: Vista del mecanismo del selector ensamblado .....                                  | 78  |
| Figura 82: Montaje de la cámara .....   | 79  |
| Figura 83: Visión general de la electrónica montada.....                                      | 79  |
| Figura 84: Esquema de conexiones y comunicaciones del Jetson Nano .....                       | 80  |
| Figura 85: Esquema de conexiones del Jetson Nano .....  | 81  |
| Figura 86: Esquema de conexiones del Arduino Mega .....                                       | 82  |
| Figura 87: Esquema de conexiones del ESP32.....   | 84  |
| Figura 88: Diagrama de flujo del programa principal del Jetson Nano .....                     | 85  |
| Figura 89: Diagrama de flujo del programa del Arduino Mega .....                              | 87  |
| Figura 90: Diagrama de flujo del proceso para girar el selector.....                          | 88  |
| Figura 91: Diagrama de flujo del proceso para bajar las compuertas.....                       | 89  |
| Figura 92: Diagrama de flujo del proceso para subir las compuertas .....                      | 90  |
| Figura 93: Diagrama de flujo del proceso para realizar el homing del selector .....           | 91  |
| Figura 94: Esquema del proceso de captación de los niveles, transmisión y visualización ..... | 92  |
| Figura 95: Diagrama de flujo del programa del emisor LoRa .....                               | 93  |
| Figura 96: Emisor LoRa. Importación de librerías .....  | 94  |
| Figura 97: Emisor LoRa. Definición de variables .....   | 94  |
| Figura 98: Emisor LoRa. Definición de variables .....   | 95  |
| Figura 99: Emisor LoRa. Inicialización de la OLED .....                                       | 96  |
| Figura 100: Emisor LoRa. Función de inicialización del LoRa .....                             | 96  |
| Figura 101: Emisor LoRa. Función de medición del nivel .....                                  | 97  |
| Figura 102: Emisor LoRa. Función de actualización de variables .....                          | 97  |
| Figura 103: Emisor LoRa. Función para emitir el paquete de datos mediante LoRa .....          | 98  |
| Figura 104: Emisor LoRa. Función Setup .....  | 99  |
| Figura 105: Emisor LoRa. Función Loop .....   | 100 |
| Figura 106: Diagrama de flujo del programa del receptor LoRa.....                             | 101 |
| Figura 107: Receptor LoRa. Importación de librerias .....                                     | 102 |
| Figura 108: Receptor LoRa. Definición de variables .....                                      | 103 |
| Figura 109: Receptor LoRa. Definición de variables .....                                      | 103 |
| Figura 110: Receptor LoRa. Definición de variables .....                                      | 103 |
| Figura 111: Receptor LoRa. Definición de variables .....                                      | 103 |
| Figura 112: Receptor LoRa. Definición de variables .....                                      | 104 |
| Figura 113: Receptor LoRa. Definición de variables .....                                      | 104 |
| Figura 114: Receptor LoRa. Definición de variables .....                                      | 104 |
| Figura 115: Receptor LoRa. Función inserta valores en la web .....                            | 105 |
| Figura 116: Receptor LoRa. Función de inicialización dela OLED .....                          | 105 |

|  |     |
|--|-----|
| Figura 117: Receptor LoRa. Función de inicialización del LoRa .....                  | 106 |
| Figura 118: Receptor LoRa. Función de conexión WiFi.....                             | 107 |
| Figura 119: Receptor LoRa. Función de recepción del paquete y extracción de datos .. | 108 |
| Figura 120: Receptor LoRa. Función de obtención de la fecha y hora.....              | 108 |
| Figura 121: Receptor LoRa. Función Setup .....                                       | 110 |
| Figura 122: Receptor LoRa. Función Loop .....  | 111 |
| Figura 123: Diagrama de flujo del servidor web .....                                 | 112 |
| Figura 124: Servidor web. Cabecera.....  | 113 |
| Figura 125: Servidor web. Cuerpo .....   | 114 |
| Figura 126: Vista del diseño por HTML.....   | 115 |
| Figura 127: Pantalla inicial de la aplicación móvil.....                             | 116 |
| Figura 128: Pantalla de visualización de los datos en la aplicación móvil.....       | 117 |

## ÍNDICE DE ECUACIONES

|   |    |
|---|----|
| Ecuación 1: Actualización de pesos de un perceptrón.....  | 17 |
| Ecuación 2: Entropía Cruzada .....  | 19 |
| Ecuación 3: Entropia cruzada categórica .....   | 20 |
| Ecuación 4: Entropía cruzada categórica simplificada a problemas multiclase .....   | 20 |
| Ecuación 5: Cálculo de las redivadas parciales de la función de coste .....   | 21 |
| Ecuación 6: Actualización de los pesos, mediante el descenso del gradiente .....  | 21 |
| Ecuación 7: Actualización de los pesos, mediante optimizador SGD .....  | 23 |
| Ecuación 8: Actualización de los pesos, mediante el optimizador Momentum .....  | 25 |
| Ecuación 9: Retención o media exponencial de los cuadrados de las componentes de los<br>gradientes en el optimizador RMSProp.....                         | 26 |
| Ecuación 10: Tamaño del paso en el optimizador RMSProp .....  | 26 |
| Ecuación 11: Actualización de los pesos con el optimizador RMSProp .....  | 26 |
| Ecuación 12: Retención o media exponencial de las componentes de los gradientes en el<br>optimizador ADAM (al igual que en SGD).....                      | 27 |
| Ecuación 13: Retención o media exponencial de los cuadrados de las componentes de los<br>gradientes en el optimizador ADAM (al igual que en RMSprop)..... | 27 |
| Ecuación 14: Tamaño de los pasos en el optimizador ADAM .....   | 27 |
| Ecuación 15: Actualización de los pasos con el optimizador ADAM .....   | 27 |
| Ecuación 16: Actualización de los pesos con el optimizador NADAM .....  | 28 |
| Ecuación 17: Función exponencial normalizada .....  | 35 |
| Ecuación 18: Cálculo de la distancia entre el eje de la compuerta cerrada (q1) y el eje del<br>émbolo (q3) .....  | 46 |
| Ecuación 19: Distancia vertical entre eje q2 y eje q3 (compuertas cerradas).....  | 46 |
| Ecuación 20: Cálculo de la longitud del eslabón .....   | 46 |
| Ecuación 21: Cálculo de la distancia entre el eje de la compuerta abierta (q1) y el eje del<br>émbolo (q3) .....  | 48 |
| Ecuación 22: Cálculo de componente c .....  | 48 |
| Ecuación 23: Cálculo de componente b .....  | 48 |
| Ecuación 24: Cálculo de componente a .....  | 48 |
| Ecuación 25: Cálculo de componente x .....  | 48 |
| Ecuación 26: Cálculo de componente y .....  | 48 |
| Ecuación 27: Transformación del tiempo de propagación de la onda del sensor de<br>ultrasonido a distancia .....   | 97 |

## ÍNDICE DE TABLAS

|   |     |
|---|-----|
| Tabla 1: Resultados obtenidos de los experimentos realizados con diferentes optimizadores ..... | 41  |
| Tabla 2: Parámetros y tiempos de impresión .....  | 72  |
| Tabla 3: Cálculo de costes .....  | 118 |

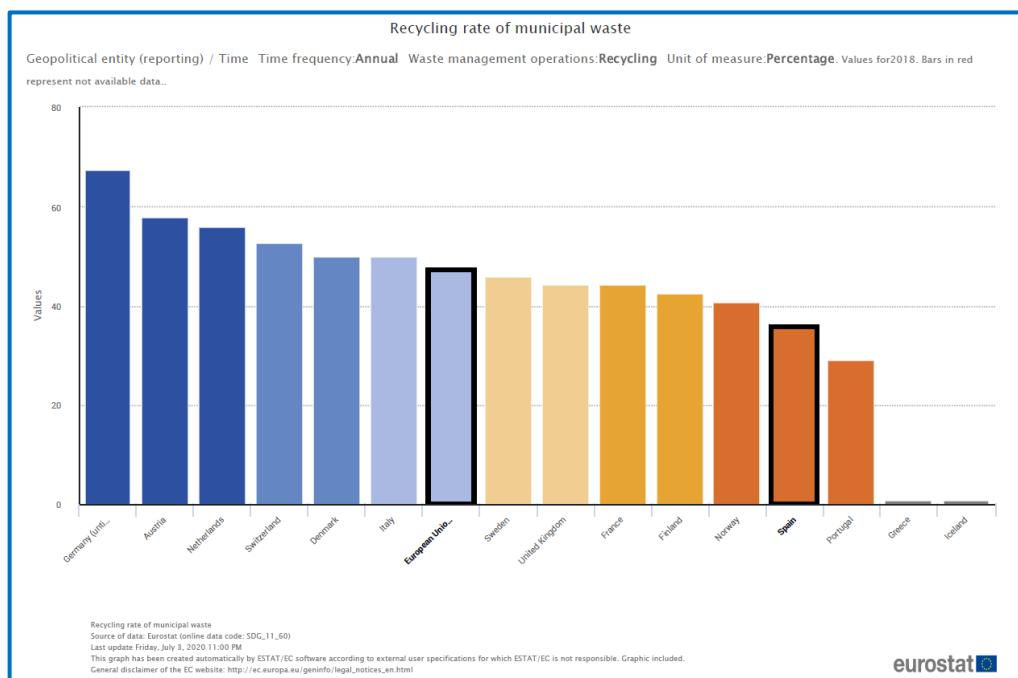
## 1. INTRODUCCIÓN

### 1.1. Planteamiento del problema

La crisis climática y ecológica continúa avanzando y el ser humano se enfrenta a un desafío sin precedentes para el cual deberá desplegar todo su ingenio para minimizar y eliminar sus devastadoras consecuencias.

La gestión de los residuos es uno de los grandes retos que debe ser abordado. En primer lugar, reduciendo en la medida de lo posible los residuos no biodegradables o con imposibilidad de ser reciclados o reutilizados y, por otra parte, mejorando las técnicas de clasificación y gestión de los mismos.

En un país desarrollado como España, en el año 2018 se recicló solamente el 36% de los residuos urbanos (*Figura 1*), y por debajo de la media europea que apenas alcanza el 50%. Si se analizan datos más globales, como la media de los países miembros de la OCDE, son aún más desmoralizadores, ya que solamente se recicló un 26% de los residuos en el año 2018 (fuente: OECD).



**Figura 1:** Niveles de reciclaje en países de la Unión Europea.

**Fuente:** Gráfica extraída de la página web de Eurostat (1)

Una gestión ineficiente de los residuos, implica daños medioambientales tanto directos, como la deposición final de estos en ríos o mares afectando a sus ecosistemas, como indirectos, provocando un aumento de las emisiones necesarias para fabricar de nuevo dichos productos, incurriendo también en un mayor coste de producción.

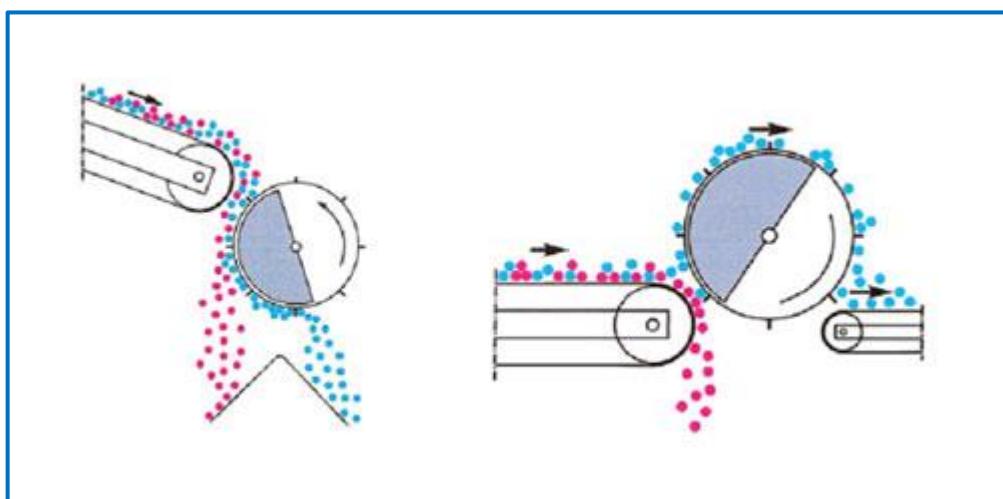
El proceso de transporte de los residuos es también una etapa, en la mayoría de los casos, contaminante pero con un alto grado de optimización, ya que las rutas de recogida de los residuos son planificadas, en el mejor de los casos, en función de datos estadísticos.

Entre las causas de estos bajos niveles de reciclaje se encuentran la poca concienciación o preocupación por parte de los ciudadanos, ya sea porque los estados o las organizaciones competentes no han sabido comunicar correctamente esta problemática, o porque simplemente carecen de la suficiente información que proporcionan los propios residuos. En este aspecto, a nivel global existe un potencial de explotación de esta información bastante relevante.

## 1.2. Precedentes y otras alternativas

La clasificación de residuos es un problema hasta la fecha difícil de automatizar dada la variedad de colores, formas y texturas de los residuos que se generan en una sociedad moderna. Las tecnologías de visión artificial convencionales son incapaces de resolver de una forma eficaz este problema.

Mediante el uso de sensores inductivos o electroimanes se pueden separar los residuos metálicos. Por ejemplo, empleando tambores separadores que cuentan con una mitad magnética para recoger los elementos metálicos mezclados con el resto de los residuos y otra mitad no magnética para depositarlos en otro lugar separado. (*Figura 2*).



**Figura 2:** Tambores separadores magnéticos

**Fuente:** Imagen extraída de la web de interempresas (2)

En efecto, el uso de sensores inductivos es uno de los métodos para separar los residuos metálicos, pero existen otros, como el que proponen Mejía et al. en su proyecto *Recolección y clasificación automática de desechos reciclables* (3), que se basa en la utilización de sensores capacitivos para clasificar diferentes tipos de residuos, no por su forma, sino por su composición, es decir, por su valor dieléctrico.

En la última década, el aprendizaje profundo o *Deep learning* basado en redes neuronales, ha probado ser una eficaz solución para problemas complejos que involucran estructuras complejas de datos como fotografías con redes neuronales convolucionales CNN, o secuencias temporales como videos o audios con redes neuronales recurrentes RNN.

## 2. SOLUCIÓN PROPUESTA Y JUSTIFICACIÓN

La clasificación es uno de los problemas más estudiados y básicos en el campo del *Deep Learning*. Es por ello por lo que, en este proyecto, se aprovechará esta técnica para resolver el problema de la clasificación de los residuos.

Esta técnica, que cada año cuenta con redes con un grado mayor de optimización y, por lo tanto, mayor precisión, permite, entre otras cosas, extraer información tan importante como es la visual con la ventaja, además, de poder añadirle información de otras fuentes o sensores para conseguir una solución más robusta.

También se presenta como una solución que puede estar en continua mejora y aprendizaje. Esto deviene esencial, pues las formas y colores de los residuos son totalmente variables, así como el proceso de reciclaje y su normativa que varía en el tiempo y en atención a la situación geográfica en la que se implemente.

Por otro lado, la utilización de sensores capacitivos o de otro tipo que pueda ofrecer información sobre las propiedades intrínsecas de los residuos pueden no ser del todo útiles en casos en los que el residuo se compone de varios materiales. En este proyecto se puede entrenar a la red para que, por ejemplo, detecte un brik de leche y lo clasifique como plástico, como así lo indica la normativa de reciclaje, a pesar de estar formado por polietileno, aluminio y, en su mayor parte, por cartón. La red será capaz, en este caso, de reconocer por su forma y color que se trata de un brik de leche.

### 3. RESUMEN Y OBJETIVOS

Este proyecto tiene como objetivo presentar una solución basada en redes neuronales para la clasificación de residuos en 5 clases: 1) papel o cartón; 2) plástico; 3) metal; 4) vidrio; y 5) orgánico.

Para ello se recopilará un set de imágenes de estos residuos y se entrenará a una red neuronal con una arquitectura y unos algoritmos debidamente justificados en la presente memoria.

Esta red será posteriormente implementada en un prototipo de máquina, diseñada y construida íntegramente por el autor mediante impresión 3D de modelación por deposición fundida.

Esta máquina será capaz de gestionar los movimientos necesarios para la clasificación de los residuos, la medida de los niveles de los depósitos de cada residuo, y su posterior transmisión mediante la tecnología de comunicación de bajo consumo y larga distancia LoRa, con la intención de integrar esta máquina en la denominada *Smart city* o ciudad inteligente.

Para poder gestionar todos estos subprocessos y repartirse las tareas principales, es decir, la detección del residuo e inferencia sobre la red neuronal, la gestión de los movimientos y la transmisión LoRa, se propondrá una estructura organizativa compuesta por diferentes microcontroladores y una comunicación entre los mismos que permita una fluidez adecuada entre subprocessos.

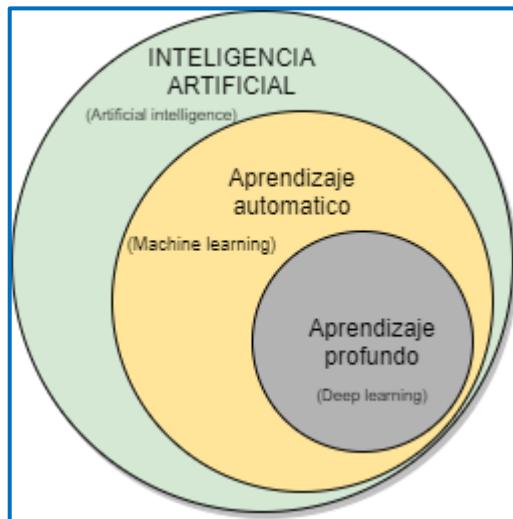
Así mismo, se propone desarrollar una aplicación móvil para dispositivos Android, para la visualización de los niveles de los residuos vertidos en cada depósito. Esto podría ser útil en el caso de que la maquina estuviese hipotéticamente integrada en una red de semejantes mucho mayor y esta información podría ser explotada por una empresa de gestión de residuos. De esta forma, se podría optimizar la recogida de los residuos en función de los niveles presentes en cada depósito, reduciendo así, considerablemente, los costes y las emisiones de sus flotas de vehículos. Esta información también puede ser de especial valor para iniciar campañas de concienciación a la ciudadanía sobre los residuos que se generan.

Por otro lado, se recopilarán todas las imágenes de los residuos clasificados en una base de datos interna en la propia máquina. Estos serán clasificados en diferentes directorios según la clase que la red neuronal ha considerado que pertenecen. Esta información puede ser útil para evaluar el funcionamiento de la red neuronal.

## 4. IMPLEMENTACIÓN DE UNA RED NEURONAL

### 4.1. Principios básicos. Perceptrón

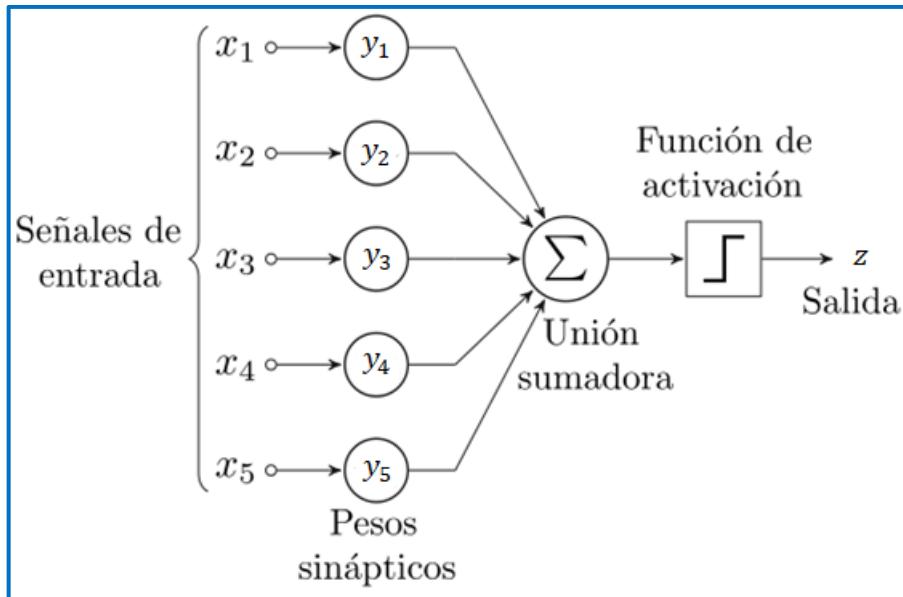
La **inteligencia artificial** (4) se compone de diferentes técnicas computacionales que permiten resolver problemas y tareas normalmente desarrolladas por la inteligencia humana. Dentro de este campo se incluye uno más delimitado, el del **aprendizaje automático**, que imita las técnicas de aprendizaje a base de *prueba y error*, método empleado por la naturaleza a lo largo de millones de años para que los seres vivos fuesen capaces de desarrollar exitosamente tareas cada vez más complejas.



**Figura 3:** Diagrama inteligencia artificial, aprendizaje automático y profundo

**Fuente:** Elaboración propia

El **aprendizaje profundo** consiste en la simulación de redes neuronales como las de un cerebro animal. En este caso, la neurona artificial se denomina **perceptrón** (*Figura 4*) (5), que no es más que el modelado matemático de una neurona. Este algoritmo consiste en una función matemática de suma ponderada que multiplica las entradas  $x_i$  por un peso  $y_i$ , suma cada uno de los resultados y los divide por el número de entradas. Este resultado es comparado con el esperado y la diferencia entre ellos ajusta el valor de los pesos.



**Figura 4:** Estructura de un perceptrón

**Fuente:** Elaboración propia

Para el proceso de aprendizaje de un perceptrón, es decir de ajuste o actualización de los pesos  $y_i$ , se emplea la siguiente fórmula:

**Ecuación 1:** Actualización de pesos de un perceptrón

$$y'_i = y_i + t * (\delta - z) * x_j$$

Donde  $y'_i$  es el nuevo peso actualizado que es igual a la suma del peso anterior  $y_i$  más la diferencia del valor deseado,  $\delta$ , respecto a la salida  $z$  multiplicada por la entrada correspondiente  $x_i$  y, en el caso de un entrenamiento con tasa de aprendizaje<sup>1</sup>, se multiplica por dicha tasa  $t$ . Con esto se consigue ir reduciendo, en cada iteración de aprendizaje, la diferencia entre el valor deseado y la salida obtenida, es decir, el perceptrón “aprende”.

Por ejemplo, y simplificadamente, en el caso de que se quisiera entrenar a sumar a un perceptrón, se procedería de la siguiente forma: 1º) se introducen las entradas, en este caso los números que se desean sumar y 2º) estos números se someten a la suma ponderada empleando para la primera iteración unos pesos generados aleatoriamente por la computadora.

Como es evidente, el resultado de dicha operación será también aleatorio en esta primera iteración. Este resultado se comparará con el correcto y la diferencia se empleará junto con la tasa de aprendizaje o en inglés *learning rate* para ajustar en mayor o menor medida los pesos. Por tanto, si esta tasa es demasiado pequeña puede demorar demasiado el proceso de aprendizaje y si es demasiado grande, puede sobrepasar el resultado esperado.

Cuando se obtenga una diferencia que se halle entre el valor obtenido y el deseado dentro del umbral requerido, se considera que la neurona está entrenada. En este caso, sería cuando los pesos son cercanos a la unidad y la red devuelve la suma de las entradas.

<sup>1</sup> Se comentará más en detalle en epígrafes posteriores

## 4.2. Principios básicos. Redes neuronales

Se entiende por red neuronal o **perceptrón multicapa** la agrupación de varias neuronas en múltiples capas.

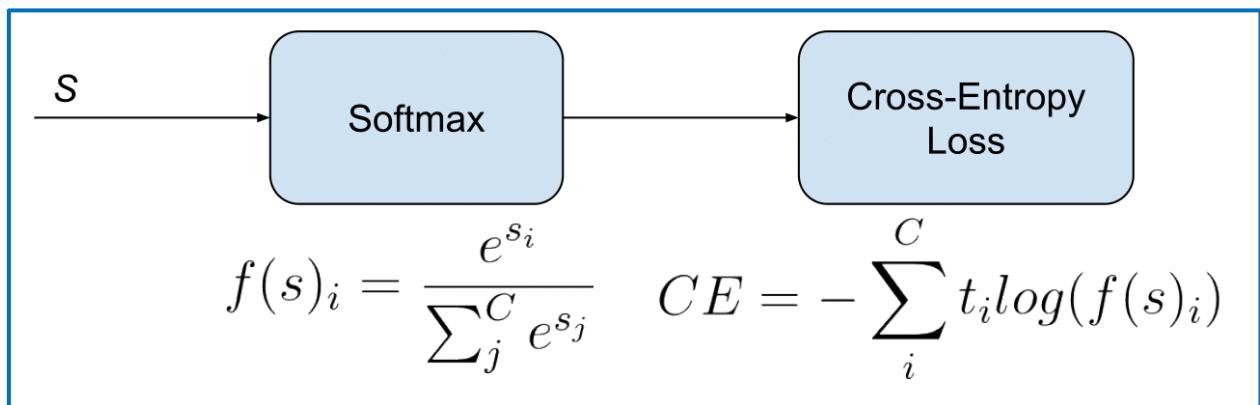
Esto es necesario para resolver problemas complejos, como acontece en el caso de este proyecto, con la clasificación a partir de imágenes.

Para problemas de visión artificial, como clasificación o segmentación de imágenes, las redes neuronales empleadas son del tipo convolucional, es decir, formadas por múltiples capas de neuronas colocadas de forma consecutiva. Esto tiene su razón de ser, al ser una analogía de la estructura de la corteza visual primaria de un cerebro animal.

### 4.2.1 Función de coste

Una red neuronal se considera que “aprende” cuando se reduce su **función de pérdida** (6) o coste (*loss function*). Es decir, esta función determina el error en el que incurre una red neuronal en función de sus parámetros o pesos.

Hay diversas opciones para calcular este error, pero en problemas de clasificación multiclase<sup>2</sup> es común emplear una **función de entropía cruzada categórica** (*categorical cross-entropy loss*), que consiste en una función de entropía cruzada o también denominada logarítmica, precedida por una capa densa con función de activación de tipo **Softmax**<sup>3</sup>.

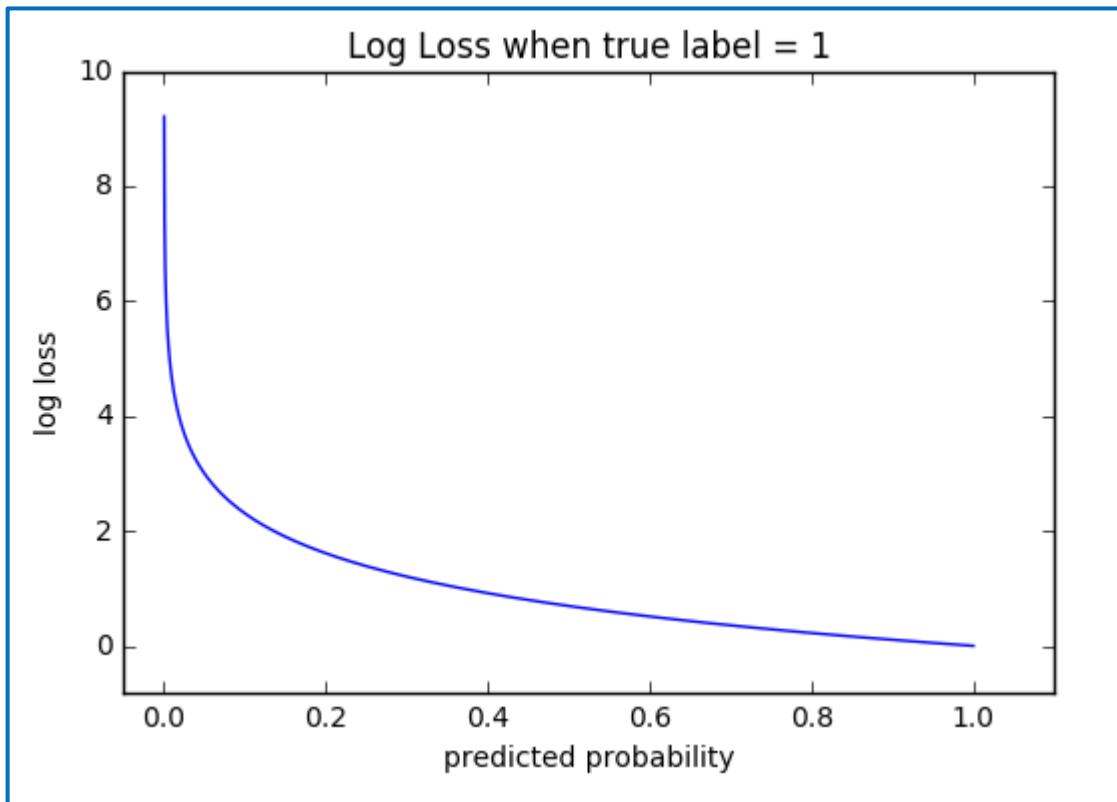


**Figura 5:** Función de entropía cruzada categórica  
**Fuente:** Imagen extraída del blog de Raúl Gómez (7)

<sup>2</sup> Un problema multi-clase es aquel en el que los ejemplos pueden pertenecer a una sola clase a la vez, en contraposición a multi-etiqueta que podrían pertenecer a varias clases. Por ejemplo, considerando como clases “mamífero” y “marino”, una ballena podría pertenecer a las dos y, por tanto, se trataría como un problema multi-etiqueta.

<sup>3</sup> Este tipo de capa se explicará en el epígrafe 5.2.3, en el que se desarrolla la construcción de la red

La función de pérdida por entropía cruzada devuelve una probabilidad entre 0 y 1 de que lo que se está analizando forme parte de esa clase y disminuye de manera logarítmica según el resultado se aproxime al real.



**Figura 6:** Gráfica de la pérdida de una función de entropía cruzada respecto a que la probabilidad predecida sea la correcta

**Fuente:** Imagen extraída de la plataforma GitHub (8)

Este tipo de función de pérdida viene determinada por la siguiente fórmula:

**Ecuación 2:** Entropía Cruzada

$$EC = - \sum_i^c t_i \log (S(y)_i)$$

Siendo:

$EC$ : la función de pérdida por entropía cruzada

$t_i$ : es un valor booleano 0 o 1 que indica si verdaderamente el ejemplo que se está procesando pertenece a la clase  $i$ , esto es un dato conocido ya que es un proceso de **entrenamiento supervisado**, es decir, cada imagen que se procesa está etiquetada con su clase correcta.

$S(y)_i$ : es la función de activación, en este caso de tipo Softmax para cada una de las clases, es decir la probabilidad calculada por la red de que pertenezca a la clase  $i$ .

Al estar precedida la función de entropía cruzada por una función de tipo Softmax, sustituyendo en la ecuación anterior resulta:

**Ecuación 3:** Entropía cruzada categórica

$$ECC = - \sum_i^c t_i \log \left( \frac{e^{y_i}}{\sum_{k=1}^K e^{y_k}} \right)$$

*ECC*: función de pérdida por entropía cruzada categórica

Al tratarse de un problema multiclase y no multi-etiqueta, solo una de las clases será la correcta (*one-hot*) así que solo un  $t_i$  será 1 y por tanto la fórmula se puede reducir a lo siguiente:

**Ecuación 4:** Entropía cruzada categórica simplificada a problemas multiclase

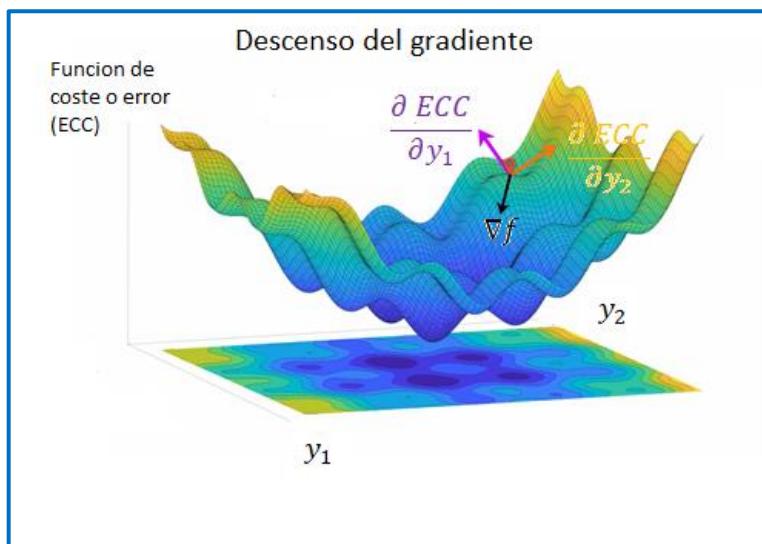
$$ECC = -\log \left( \frac{e^{y_i}}{\sum_{k=1}^K e^{y_k}} \right)$$

Donde

$e^{y_i}$ : es la probabilidad calculada para la clase correcta

#### 4.2.2. Descenso del gradiente y optimizadores

Para reducir esta función de pérdida o el error a un mínimo se efectúa lo que se denomina método de descenso del gradiente. Para ello es necesario calcular el gradiente de la función de pérdida.



**Figura 7:** Interpretación del descenso del gradiente

Fuente: Elaboración propia a partir de un vídeo divulgativo de gradientes (9)

En la *Figura 7* se representa lo que podría ser una función de coste o pérdida, en este caso de tipo entropía cruzada categórica respecto solamente a 2 parámetros o pesos ( $y_1$  y  $y_2$ ). En el caso de la red de este proyecto se actualizan 5010 parámetros<sup>4</sup>, es decir que si para este ejemplo se necesita una gráfica en 3 dimensiones para el proyecto sería de 5011 dimensiones.

Se calculan por tanto las derivadas parciales de la función de coste o error respecto a cada uno de los parámetros entrenables de la red y se obtiene así el gradiente que se debe restar para descender hasta un “valle” de dicha función.

<sup>4</sup> Como se justificará en el epígrafe 5.2.3, donde se desarrolla la construcción de la red

En este proyecto se emplea la función de coste de tipo ECC y, por lo tanto, las derivadas parciales se calcularían según la función:

**Ecuación 5:** Cálculo de las rederivadas parciales de la función de coste

$$\nabla * \sum_0^m ECC = [\frac{\partial}{\partial y_i} \left( -\log \left( \frac{e^{y_i}}{\sum_{k=1}^K e^{y_k}} \right) \right)] = \left( \frac{e^{y_i}}{\sum_{k=1}^K e^{y_k}} \right) - 1$$

Siendo  $y_i$  cada uno de los 5010 parámetros o pesos entrenables.

Una vez conocido el gradiente será necesario poder calcular el nuevo valor al que se deben de actualizar cada uno de los pesos, para descender hasta un ‘valle’ de la función de coste.

Es decir, se aplica un algoritmo de optimización que permite converger hacia un valor mínimo de la función de pérdida mediante un proceso iterativo. Cada iteración corresponde a lo que se denomina **paso** o *step*. Para este proceso es necesario definir la **tasa de aprendizaje** o *machine learning*, que consiste en establecer en qué medida se actualizan los parámetros de la red en cada iteración, es decir, de forma metafórica, el tamaño de los pasos que se ejecutan para descender hacia un “valle” de la función de pérdida siguiendo la dirección de descenso del gradiente. Por tanto, como se comentó en el epígrafe 4.1, si el valor es muy grande, el proceso de alcanzar un mínimo puede ser impracticable e impreciso, pero, por otro lado, si es demasiado pequeño, el proceso de entrenamiento podría demorarse demasiado. La obtención de estos hiperparámetros es, en la mayoría de los casos, a base de pruebas y comparativas.

Por cada paso que se da, se introducen en la red múltiples ejemplos, en este caso múltiples imágenes. La cantidad de estas se denomina **tamaño de lote** o *batch size*. En este término se profundizará más adelante.

Con todo esto se puede definir la ecuación básica que determina cómo se actualizan los pesos en cada paso:

**Ecuación 6:** Actualización de los pesos, mediante el descenso del gradiente

$$y'_i = y_i - t * \nabla * \sum_0^m ECC$$

Donde:

$y'_i$  es el peso actualizado después del paso

$t$  : es la tasa de aprendizaje

$\nabla$  : es el gradiente respecto a los pesos

$\sum_0^m ECC$ : es la suma de todas las funciones de pérdida para cada imagen o muestra  $m$  (en este caso la función es de tipo Entropía Cruzada Categórica).

De esta ecuación se puede deducir que el signo negativo se debe a que se quiere descender en la dirección opuesta al gradiente y que el tamaño de los pasos, o la forma de actualizar los pesos, no depende en realidad solamente de la tasa de aprendizaje sino también del gradiente. Por tanto, cuando se está acercando a un mínimo de la función, el gradiente disminuirá también y por ende el tamaño de los pasos, esto es ideal para que conforme se

acerque al mínimo de la función no se sobrepase (*overshoot*). También cabe comentar que las diferentes funciones de actualización de los pesos que se irán comentando se realizan desde el final de la red hacia adelante (*backpropagation*). Es decir, los pesos finales son los más susceptibles a ser modificados y en cambio los del principio, menos. Tiene su razón de ser puesto que son realmente las primeras capas de la red las encargadas de reconocer rasgos más primitivos o simples y las últimas, en función de las primeras, determinan rasgos más complejos y son por tanto las que más relevancia tienen en el “veredicto” final.

Conviene también que la tasa de aprendizaje de esta función se vaya reduciendo con cada iteración a medida que se desciende hasta el mínimo. Por ello existen diferentes optimizadores (10) que lo van adaptando.

Por otro lado, es necesario enfrentarse a otro problema a la hora de alcanzar el mínimo, y es que en las funciones de coste reales no existe un solo mínimo de la función sino múltiples, la función nunca va a ser puramente convexa, sino que tiene una geometría totalmente irregular con máximos y mínimos locales y globales. Por ello, es necesario alcanzar la combinación de los pesos que permita obtener un mínimo global y no local.

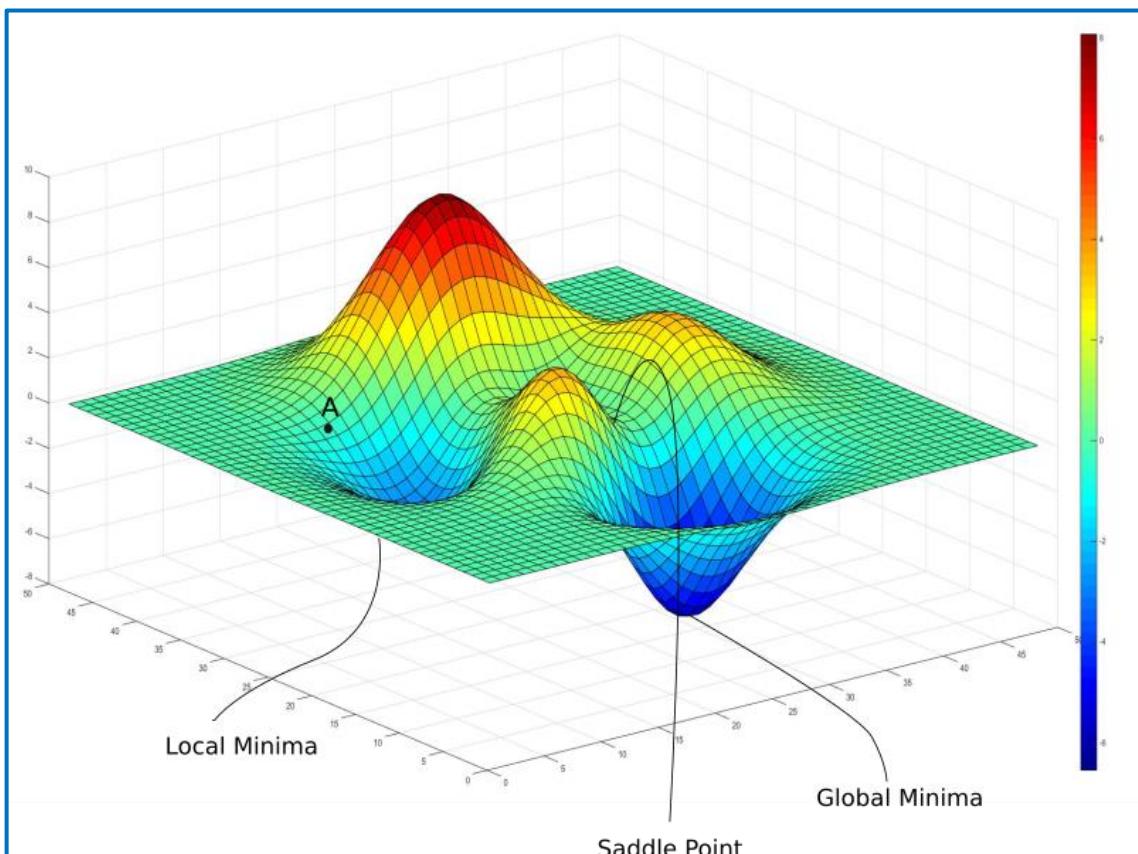


Figura 8: Punto de silla de caballo

Fuente: Imagen extraída del blog paperspace.com (11)

Además, como se muestra en la *Figura 7* se puede alcanzar un Punto de Silla de Caballo o *Saddle Point* que es un mínimo en una dirección, pero un máximo en la contraria. Con la ecuación anterior de optimización de los pesos se quedaría estancada en este punto o en algún mínimo local. Por tanto, es necesario introducir el algoritmo de optimización de tipo **SGD** (*Stochastic Gradient Descent*) o descenso del gradiente estocástico (12).

Como su nombre indica se recurre a la aleatoriedad para realizar el cálculo del gradiente. En este caso no se realizará la suma de todas las funciones de pérdida de todas las imágenes introducidas para calcular dicho gradiente sino solo una de ellas de forma aleatoria hasta pasar por todas las imágenes. Por tanto, la ecuación de optimización pasaría a ser:

**Ecuación 7:** Actualización de los pesos, mediante optimizador SGD

$$\mathbf{y}'_i = \mathbf{y}_i - \mathbf{t} * \nabla * \mathbf{ECC}_a$$

Donde

$\mathbf{ECC}_a$ : ya no es el sumatorio de las funciones de pérdida para cada muestra sino una función de pérdida de una muestra tomada al azar.

Por tanto, el SGD se contrapone al BGD (*Batch Gradient Descent*) definido en la ecuación 5. El BGD es computacionalmente inviable al tener que sumar el gradiente de las funciones de coste generadas al introducir todas las imágenes del set de entrenamiento.

Existe un punto intermedio entre el SGD y el BGD y se denomina *mini-batch SGD* ya que recurre a *mini-batch*, es decir, grupos de imágenes de un tamaño determinado (*Batch Size*) formado a partir de imágenes seleccionadas de forma aleatoria del set de imágenes destinadas al entrenamiento, normalmente compuesto por 16, 32 o 128 muestras. En cada paso se introduce este grupo de imágenes y se calcula el gradiente a partir de la suma de las funciones de pérdida generadas por cada una de las imágenes del grupo.

No obstante, al optimizador SGD se le puede presentar un nuevo obstáculo, esta vez con forma de desfiladero lo que en inglés se conoce como *pathological curvature*. Consiste en una curva muy pronunciada en la función de coste en una dirección y menos pronunciada en la contraria. O, lo que es lo mismo, el gradiente en un sentido es mucho mayor que en el contrario.

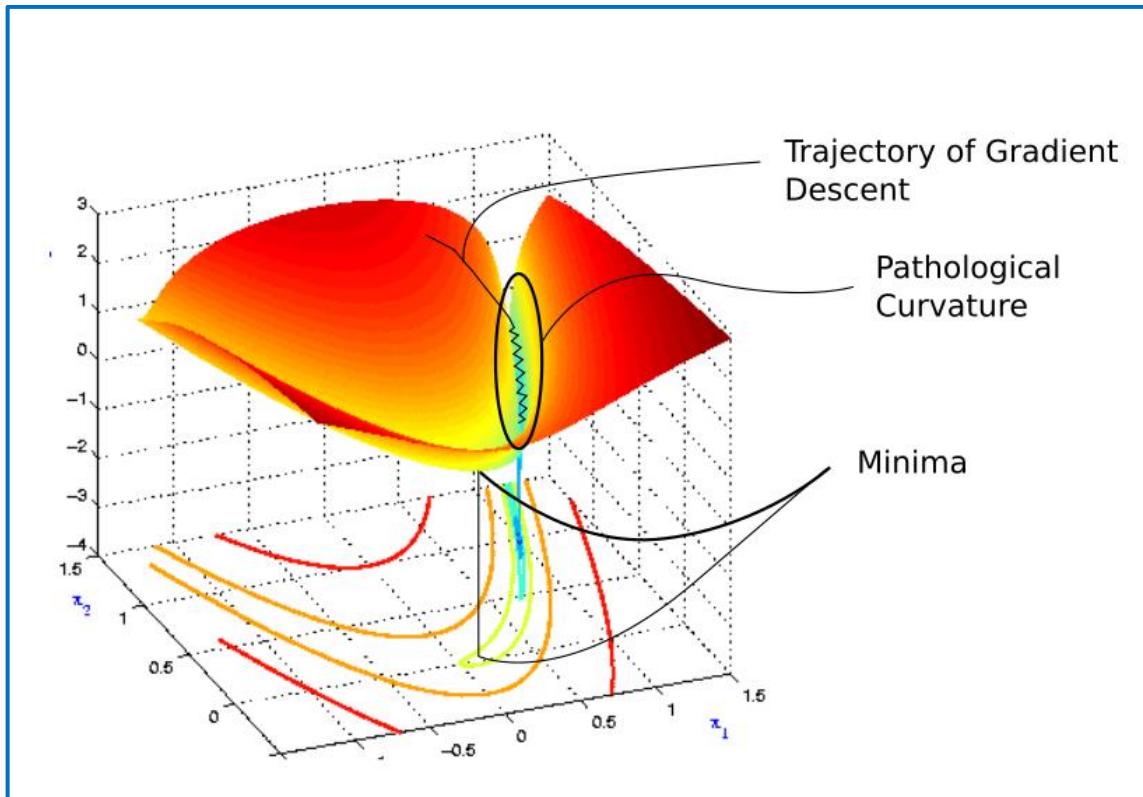


Figura 9: Curva patológica

Fuente: Imagen extraída del blog paperspace.com (11)

Por tanto, empleando por ejemplo el SGD se demoraría mucho en descender hacia el mínimo porque seguiría una trayectoria en zigzag como la siguiente:

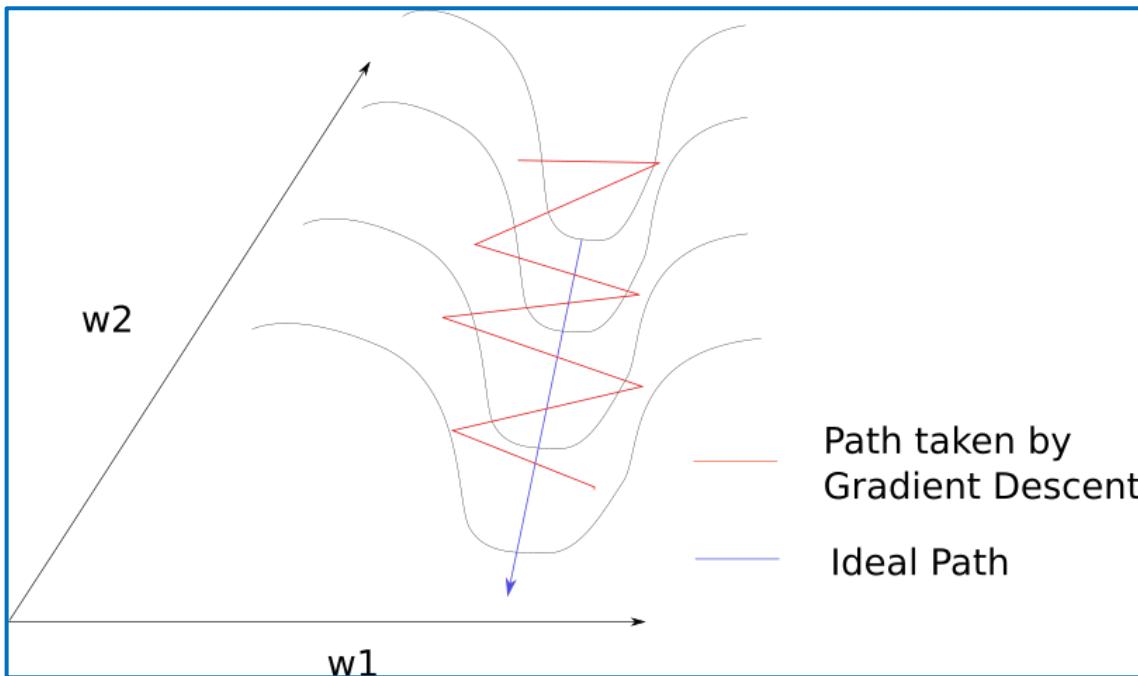


Figura 10: Trayectoria de descenso sobre una curva patológica

Fuente: Imagen extraída del blog paperspace.com (11)

Es necesario lograr un método que disminuya la velocidad (el tamaño de los pasos) mientras se descienda al fondo del desfiladero y que la aumente una vez en el fondo, en

dirección al mínimo. Una herramienta matemática que puede ser útil para esto son las segundas derivadas.

El descenso del gradiente, como se observó, solo emplea la primera derivada para determinar si la función de coste está disminuyendo y en qué medida lo hace. Pero no es capaz de reconocer la forma de la curva, es decir si es convexa, cóncava, o simplemente plana. Para conocer estos aspectos es necesario recurrir, por tanto, a las segundas derivadas.

Dentro de este grupo, denominado **métodos adaptativos** porque sí que tienen en cuenta la forma de la curva y no solo su gradiente para adaptar el tamaño de los pasos, encontramos el **método de Newton** que ajusta dicho tamaño siguiendo, claro está, el camino descendente del gradiente. Para ajustar los pasos, calcula la matriz Hessiana compuesta por dobles derivadas de la función de coste respecto a todas las combinaciones de los pesos que se están entrenando. Es decir, computacionalmente impracticable, tratándose normalmente de millones de parámetros que combinar.

Otra técnica que sí puede ser empleada para actualizar los parámetros es la del **Momentum**, que es utilizada junto a la SGD y que consiste en emplear no solo el valor del gradiente actual sino también el obtenido en los pasos precedentes. Es necesario definir, por tanto, un coeficiente de momento que determine el porcentaje de retención del gradiente en cada iteración. Cuanto más recientes sean los gradientes más relevancia tendrán, de hecho, la retención será una media exponencial, es decir, la relevancia que tengan los gradientes crecerá de forma exponencial según sean más recientes. Este coeficiente suele comenzar en 0,5 subiendo con cada época hasta 0,9.

Con este optimizador la actualización de los pesos seguiría la siguiente fórmula:

**Ecuación 8:** Actualización de los pesos, mediante el optimizador Momentum

$$y'_i = y_i + m * r_i - t * \nabla * ECC_a$$

Donde:

$y'_i$ : es el peso actualizado

$y'_i$ : el valor del peso antes de actualizar

$m$ : es el coeficiente del Momentum

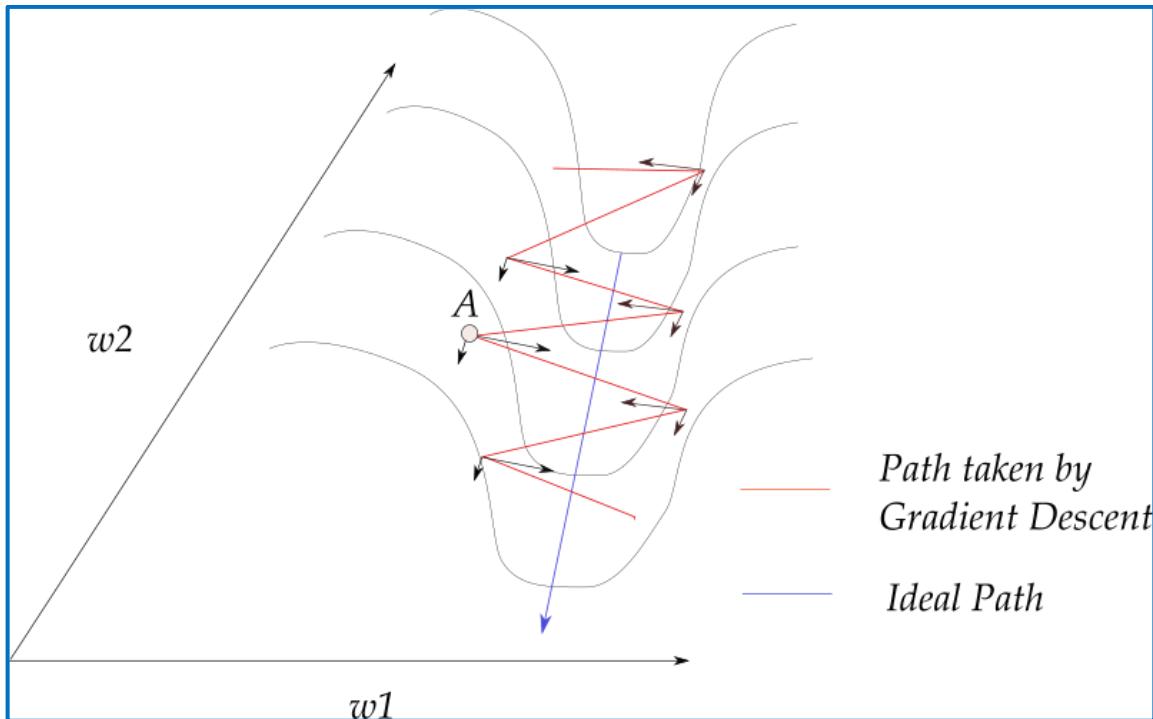
$r_i$ : son los gradientes retenidos en forma de media exponencial

$t$ : es la tasa de aprendizaje

$\nabla$ : gradiente respecto a los pesos

$ECC_a$ : función de coste que en este caso se emplea el de Entropía Cruzada Categórica

Tener en cuenta los valores anteriores de los gradientes soluciona justamente la trayectoria en zigzag que se veía en la *Figura 9*. Porque, como se puede ver en la siguiente *Figura 10*, los componentes de los gradientes en el sentido de  $w_1$  se anulan y, en cambio, los del sentido en  $w_2$  se suman, de forma que se consigue convertirlo en una trayectoria recta y acelera la convergencia hacia el mínimo, evitando oscilaciones.



**Figura 11:** Componentes del gradiente sobre las trayectorias de zigzag en una curva patológica  
**Fuente:** Imagen extraída del blog paperspace.com (11)

El optimizador de tipo **RMSProp** (*Root Mean Square Propagation*) (13) pretende reducir las oscilaciones, pero incidiendo sobre la tasa de aprendizaje, la actualiza de forma automática para cada uno de los parámetros que debe de ajustar. Este optimizador permite así guiar el descenso y evitar ir en la dirección de las oscilaciones.

Las fórmulas que se siguen para cada peso son las siguientes:

**Ecuación 9:** Retención o media exponencial de los cuadrados de los componentes de los gradientes en el optimizador RMSProp

$$c_{t_i} = c_{t-1_i} * \rho + (1 - \rho) * g_t^2$$

**Ecuación 10:** Tamaño del paso en el optimizador RMSProp

$$\Delta y_{t_i} = -\frac{t_0}{\sqrt{c_{t_i} + \epsilon}} * g_t$$

**Ecuación 11:** Actualización de los pesos con el optimizador RMSProp

$$y_{t+1_i} = y_{t_i} + \Delta y_{t_i}$$

Siendo para cada peso  $i$ :

$c_{t_i}$ : retención o media exponencial de los cuadrados de los componentes de los gradientes

$c_{t-1_i}$ : la retención de los gradientes anteriores en forma de media exponencial de los cuadrados

$\rho$ : hiperparámetro

$g_t$ : componente del gradiente en el momento  $t$  en la dirección del peso  $i$

$\Delta y_{t_i}$ : determina el tamaño de pasos, la variación que va a sufrir el peso  $i$  actualizado

$t_0$ : es la tasa de aprendizaje inicial

$\epsilon$ : este parámetro sirve para que nunca se divida por 0, suele ser  $1^{-10}$

$y_{t+1_i}$ : es el peso actualizado

$y_{t+1_i}$ : es el peso sin actualizar

Como se puede ver en la *Ecuación 10* el paso será de mayor tamaño si la retención de sus gradientes es menor. Por ejemplo, en el caso de encontrarse en el desfiladero de la *Figura 10*, el gradiente o la pendiente en la dirección del peso 1 es mucho mayor que en la del peso 2, sin embargo, se necesita que el paso sea mucho mayor en el sentido del peso 2. Con esta misma ecuación se consigue además no sobrepasar el mínimo, reduciendo el tamaño de los pasos según sea el gradiente menor.

Empleando esos dos tipos de optimizadores, el RMSProp y el SGD+Momentum se obtiene un optimizador denominado **Adam** (*Adaptive Moment Optimization*) (14), que, de una forma resumida, emplea las estimaciones del momento para adaptar la tasa de aprendizaje para cada parámetro que se está entrenando.

Se utilizarían las siguientes fórmulas para la actualización de cada peso  $i$ :

**Ecuación 12:** Retención o media exponencial de las componentes de los gradientes en el optimizador ADAM (al igual que en SGD)

$$r_{t_i} = r_{t-1_i} * \beta_1 + (1 - \beta_1) * g_t$$

**Ecuación 13:** Retención o media exponencial de los cuadrados de las componentes de los gradientes en el optimizador ADAM (al igual que en RMSprop)

$$c_{t_i} = c_{t-1_i} * \beta_2 + (1 - \beta_2) * g_t^2$$

**Ecuación 14:** Tamaño de los pasos en el optimizador ADAM

$$\Delta y_{t_i} = -\frac{t_0 * r_{t_i} * g_t}{\sqrt{s_{t_i}} + \epsilon}$$

**Ecuación 15:** Actualización de los pasos con el optimizador ADAM

$$y_{t+1_i} = y_{t_i} + \Delta y_{t_i}$$

Siendo:

$\beta_1$ : un hiperparámetro, normalmente 0,9

$c_{t_i}$ : la media exponencial de los cuadrados de la componente del gradiente a lo largo del peso  $i$

$\beta_2$ : un hiperparámetro, normalmente 0,99

Como muestran estas fórmulas, para actualizar cada peso se tiene en cuenta los gradientes de cada peso individualmente como en el caso del RMSprop. Pero, en este caso, se emplea para calcular el tamaño del paso tanto la media exponencial de los gradientes,

(aportación del optimizador SGD) como la media exponencial de los cuadrados de los gradientes (aportación del RMSprop).

Por último, conviene conocer el optimizador NADAM (15). Así como el optimizador ADAM involucraba al RMSProp y el Momentum, NADAM emplea RMSProp junto al Momentum de Nesterov. Este Momentum, de una forma resumida, consiste en que, a diferencia de la técnica de Momentum que anteriormente se ha comentado, da mayor relevancia a la tasa de aprendizaje a la hora de actualizar los pesos de la red que a los gradientes retenidos.

Consiste en la siguiente formula:

**Ecuación 16:** Actualización de los pesos con el optimizador NADAM

$$y'_i = y_i + m * r_i - t * \nabla * (y_i + m * r_i) * ECC_a$$

Donde:

$y'_i$ : es el peso actualizado

$y_i$  : el valor del peso antes de actualizar

$m$  es el coeficiente del Momentum

$r_i$ : son los gradientes retenidos en forma de media exponencial (igual que la Ec. 12)

$t$ : es la tasa de aprendizaje

$\nabla$  : gradiente respecto a los pesos

$ECC_a$ : función de coste que, en este caso, se emplea el de Entropía Cruzada Categórica

## 5. RED NEURONAL

Este proyecto se podría abordar por diferentes vías. Por un lado, construir una red propia. Por otro, emplear una ya construida y entrenada para el propósito de clasificar los residuos, como podría ser la desarrollada para un proyecto similar como *Recyclenet*, proyecto desarrollado por BIRCANOGLU [et al] (16). O bien, un punto intermedio entre ambas posiciones, es decir, partir de una red ya construida especializada en clasificar y reentrenarla después con imágenes de residuos para así especializarla en resolver este tipo de clasificación. Este último proceso se conoce como *transfer-learning*.

Hay múltiples redes entrenadas específicamente sobre ingentes bases de datos y perfeccionadas, durante años, para procesos de clasificación por investigadores de todo el mundo. Por tanto, buscando una mayor fiabilidad y precisión en pro de una red propia, este proyecto empleará una red neuronal reentrenada con fotografías de residuos.

### 5.1. Set de imágenes

Para que la red neuronal se especialice en clasificar residuos es necesario alimentarla con imágenes de estos. Para ello se recurrirá a un *dataset* de Gary Thung para su proyecto *Trashnet* (17) que cuenta con 2.527 imágenes repartidas de la siguiente forma:

- Vidrio: 501
- Papel: 594
- Cartón: 403
- Plástico: 482
- Metal: 410
- Otro tipo u orgánico: 137

Para este trabajo se emplearán solo 5 grupos, descartando las imágenes pertenecientes al grupo ‘otro tipo u orgánico’ puesto que en dicho set de imágenes no existe ningún patrón común que permita clasificarlos de una forma sencilla. Con esto, para que la maquina sea capaz de clasificar residuos de tipo orgánico lo que se hará será determinar que son de este tipo cuando la inferencia resulte en un porcentaje de fiabilidad menor a un determinado número. Este extremo se concretará en epígrafes posteriores.

Además, se supplementarán estas imágenes con otras tomadas por la propia máquina que se desarrolla en este trabajo, para proporcionar a la red un fondo diferente al de las imágenes de ‘Trashnet’ unos tipos de residuos diferentes y tomadas con una cámara, la de la propia máquina que será la que finalmente cuando esté entrenada la red tenga que capturar las imágenes para su posterior clasificación.

Se han conseguido reunir:

- Vidrio: 25
- Papel: 11

- Cartón: 25
- Plástico: 37
- Metal: 22

Para agilizar este proceso se ha desarrollado un programa<sup>5</sup> en Python que permite, después de haber introducido un residuo dentro de la maquina y que por tanto la cámara lo esté enfocando, pulsar una tecla según el residuo que se haya introducido:

P→ Plástico

M→Metal

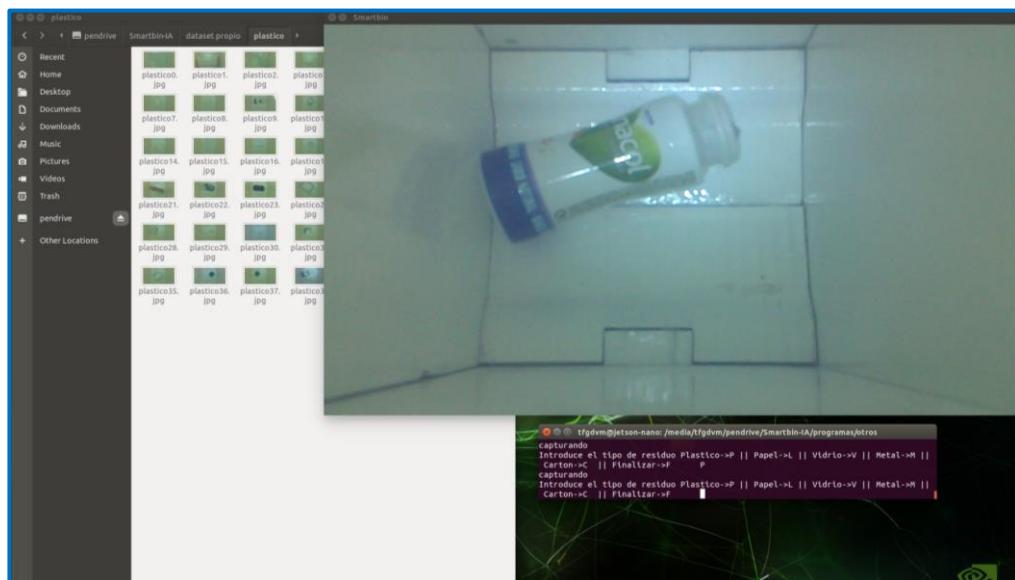
C →Cartón

L→Papel

V→Vidrio

O la tecla F si lo que se pretende es finalizar la prueba.

Pulsando una de las teclas anteriores el programa captura una imagen, la guarda en el directorio del residuo que le corresponda y le asigna un nombre con su residuo más un ID que se autoincrementa.



**Figura 12:** Proceso de creación del dataset.

**Fuente:** Elaboración Propia

Por ejemplo, al introducir el primer plástico en la máquina, se pulsaría la P y la imagen se guardaría en el directorio /pendrive(H)/Smartbin-IA/dataset propio/plástico/plastico0.jpg.

Cabe destacar que, si bien la red podría clasificar 6 tipos de residuos (incluyendo el orgánico, por descarte), la maquina separará solo 5, depositando el papel y el cartón en el mismo contenedor. Se optó por mantener el proceso de entrenamiento diferenciado en cuanto a papel

<sup>5</sup> Código completo y comentado en Anexo 15.2.2 *Código del script para generar un dataset*

y cartón porque estos materiales tienen texturas, colores y formas generalmente diferenciables y de esta forma la red logrará clasificarlos mejor si aprende a distinguirlos entre sí.

Se crearán por tanto 5 subdirectorios (papel, cartón, plástico, metal y vidrio) en una cuenta de Google Drive, y se incluirán las imágenes de este *dataset*.

## 5.2. Reentrenamiento de la red<sup>6</sup>

Para el proceso de reentrenamiento (18) se empleará la plataforma de gpu alojada Google Colab, esto permite a cualquier usuario sin una tarjeta GPU poder trabajar con redes neuronales de una forma rápida y eficaz.

### 5.2.1. Entrenamiento y validación

Normalmente en cualquier proceso de reentrenamiento de una red existen, al menos, dos etapas:

- **Entrenamiento:** durante este proceso la red neuronal ejecuta una cantidad determinada de **épocas**. En cada época, se ejecutan un numero de **pasos**, de forma que recorra todo el *dataset*. en cada uno de ellos una cantidad de imágenes (**batch size**) pertenecientes a todas las clases que se quieren clasificar son introducidas en la red para que esta determine a qué clase pertenece. Una vez determinado, al ser este un entrenamiento con entradas etiquetadas (aprendizaje supervisado), será posible ejercer un *feedback* sobre la red para que este ajuste sus pesos en función de si ha acertado o no en su resultado. Por lo tanto, en esta etapa la red “aprende”.
- **Validación:** al final de cada época se ejecuta este proceso que consiste en comprobar la precisión de la red empleando para ello un *dataset* de imágenes diferente al empleado para el entrenamiento. Cumple la función de evaluar la red y permite al programador ajustar los hiperparámetros en función de la precisión y pérdida obtenida.

Para este proyecto se realiza una división de las imágenes de forma que el 20% de ellas se procesarán durante la validación y el 80% restante se emplearán durante la etapa de entrenamiento. A veces también es habitual emplear un pequeño porcentaje del *dataset* (aproximadamente un 10%) para una tercera etapa denominada **test**, que comparte con la validación la función de evaluar, pero, en este caso, la red solo está entrenada una vez y no en cada época. Sin embargo, en este proyecto, no se empleará la etapa de test ya que se realizará en la propia maquina sobre residuos depositados en ella, y no en imágenes de este *dataset*, con ello se consigue reservar más cantidad de imágenes para una etapa prioritaria como la del entrenamiento.

### 5.2.2. Preprocesamiento de las imágenes

Partiendo del *dataset* anterior, es necesario acceder a él y realizar una serie de operaciones sobre las imágenes, con la finalidad de mejorar el proceso de reentrenamiento.

---

<sup>6</sup> Código completo y comentado en el Anexo 15.2.1 *Código de la red neuronal* y ejecutable [aquí](#)

Para ello, primero se ajustan sus dimensiones a 299x299 píxeles, pues la capa de entrada de la red que se va a reentrenar es de este tamaño. Posteriormente, y de forma aleatoria, las imágenes se rotan hasta ángulos de 40°; se desplazan horizontal y verticalmente hasta un 20% y se efectúa zum de hasta el 20% sobre ellas. Con estas modificaciones se logra obtener un *dataset* más diverso y que la red no aprenda a clasificar por la posición, desplazando las imágenes o tamaño realizando zum sobre ellas. Por ejemplo, en el caso de una lata, será capaz de reconocerla independientemente de si está boca arriba, boca abajo, cerca, o lejos de la lente.

### 5.2.3. Construcción de la red

Aunque se empleará una red ya construida es necesario añadirle unas cuantas capas más tanto a la entrada como a la salida.

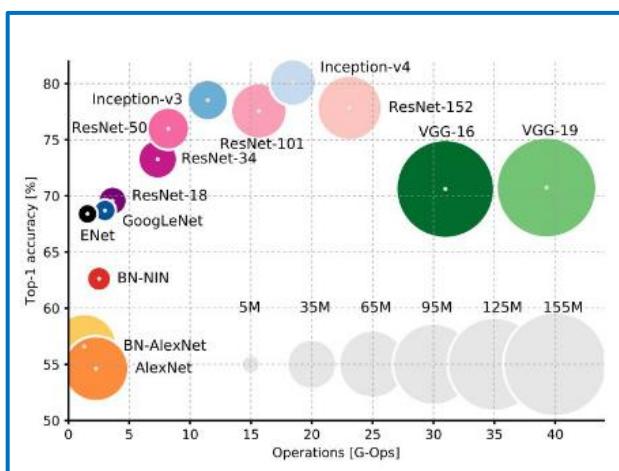
- Capa de entrada

A la entrada es necesario añadir una capa de tipo *InputLayer* con el tamaño de las imágenes 299x 299 píxeles x 3 (RGB).

- Red reentrenada

Existen múltiples redes neuronales convolucionales especializadas en clasificar que podrían ser perfectas candidatas para reentrenar en este proyecto. No obstante, se decidió optar por una arquitectura muy conocida denominada *Inception*. Su primera versión (*Inception-v1*) fue implementada por desarrolladores de Google en 2015, Szegedy [et al], de ahí que se la conozca como GoogleNet. Posteriormente se desarrolló *Inception-V2* por Ioffe y Szegedy, también en 2015, introduciendo la técnica de *batch normalization*. Y después, mediante técnicas de factorización, se obtuvo *Inception-v3* Szegedy [et al] (19).

Estas técnicas de factorización permiten reducir el número de parámetros y de conexiones sin reducir la eficiencia de la red.



**Figura 13:** Precisión red frente a operaciones de las principales redes dedicadas a clasificación  
**Fuente:** Imagen extraída de la web dataconomy (20)

Inception V3 está compuesta por aproximadamente 24 millones de parámetros, frente a los 138 millones de la red VGGNet de 2014 o Alexnet con peores resultados. Es decir, la cantidad de operaciones que debe de realizar es considerablemente menor (*Figura 10*), gracias

a las técnicas de factorización anteriormente mencionadas, e ideal por lo tanto para entrenamientos rápidos y en los que se precisen poca capacidad de computación como el que se lleva a cabo en este proyecto.

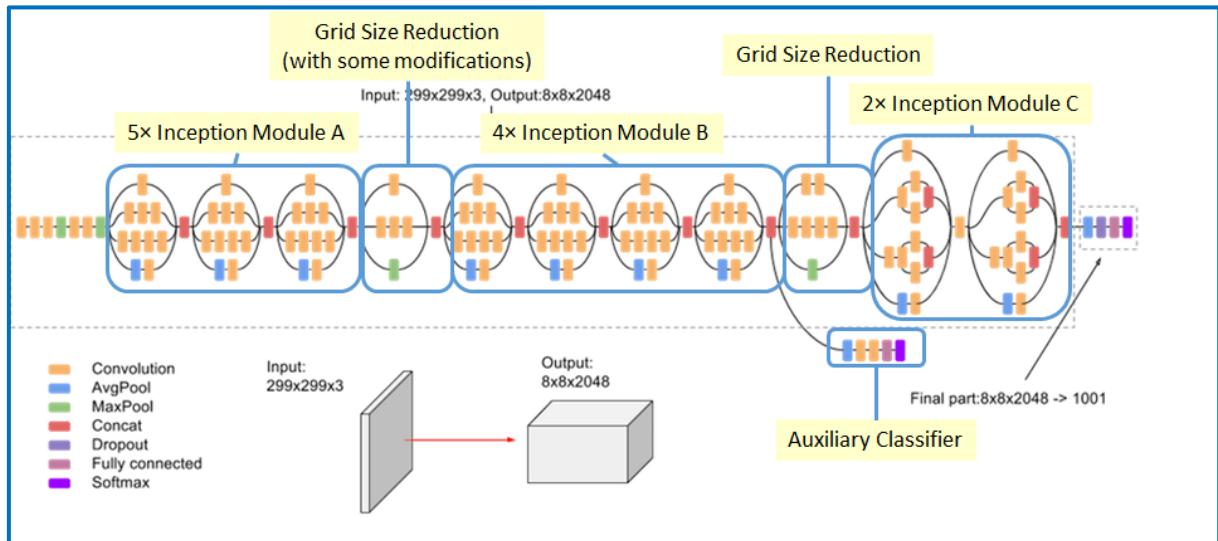
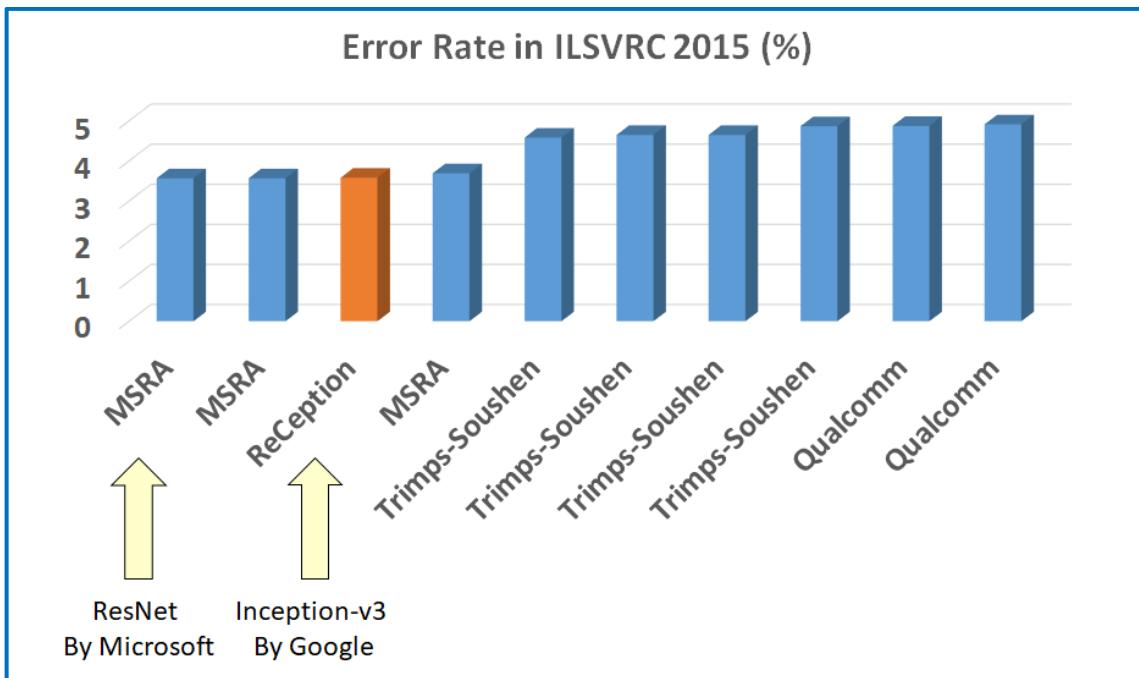


Figura 14: Arquitectura Inception V3

Fuente: Imagen extraída de la web M Colombia (21)

Inception-v3 fue entrenada sobre un subset de 1 000 000 de imágenes extraídas de ImageNet (una base de datos de más de 15 millones de imágenes y alrededor de 22 000 categorías), para poder clasificar hasta 1 000 clases diferentes, a partir de una imagen de tamaño 299x299x3.

Inception-v3, entrenado con ImageNet ILSVRC-2012-CLS (*Large Scale Visual Recognition Challenge* de 2012), un *dataset* empleado en una competición que pone a prueba anualmente a diferentes tipos de redes neuronales, logró en el certamen de 2015 unos buenos resultados como se muestra en la Figura 15.

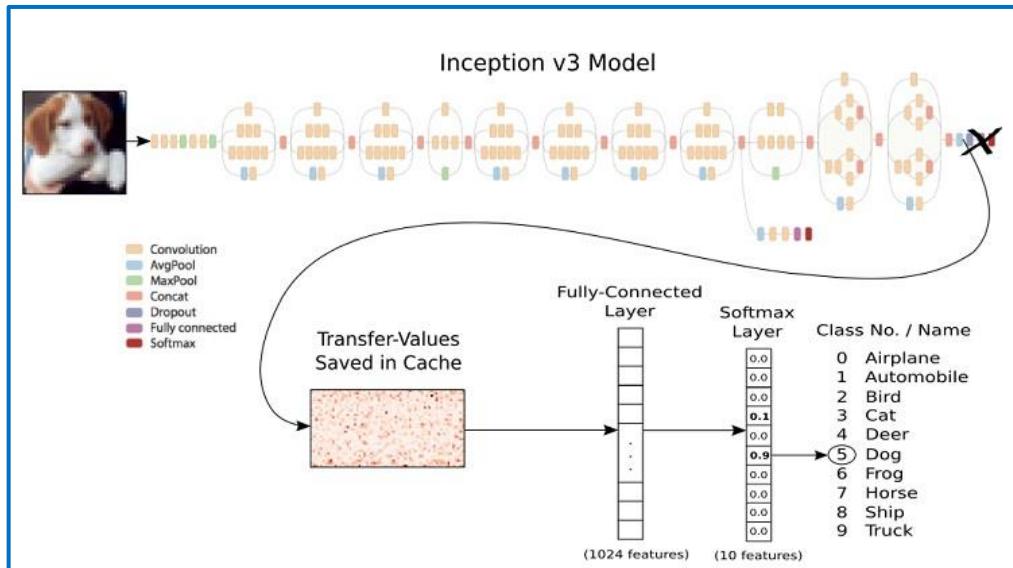


**Figura 15:** Resultados de error de diferentes redes en el certamen ILSVRC de 2015

Fuente: Imagen extraída de la web de inteligencia artificial mc.ai ([22](#))

No es de extrañar cuando en el anterior certamen su predecesora (V1) había logrado un error cercano al humano del 6.67%.

Para reentrenar una red neuronal como esta, existen dos opciones según la cantidad de pesos de la red que se ajusten: 1) reentrenar toda la red neuronal (los casi 24 millones de parámetros) o 2) reentrenar solamente las últimas capas que determinan la clase a la que pertenece la imagen de entrada (*Figura 16*).



**Figura 16:** Ejemplo de reentrenamiento de las últimas capas de Inception-v3

Fuente: Imagen extraída de la web de Intel ([23](#))

En este caso, se empleará la segunda opción pues, a diferencia de la primera, no es necesario proporcionar un *dataset* extenso (del que, por otra parte, se carece). Además, el entrenamiento es considerablemente más rápido, porque se basa en aprovechar la capacidad de

la red neuronal de reconocer ciertos rasgos primitivos en una imagen y centrarse en adaptar la clasificación final de cada clase incidiendo sobre las últimas capas. Por tanto, no se modificará ningún peso de la red de Inception-v3.

- Capa Dropout

Después de esta red, se añade una capa de tipo *Dropout*. Esta capa desactiva un porcentaje de sus neuronas de forma aleatoria, en este caso del 50%. Gracias a esta capa se evita que la red sufra el denominado efecto de *overfitting*, que consiste en que la red obtiene unos buenos resultados sobre las imágenes del set de entrenamiento ya que, en vez de aprender, memoriza y, de esta forma, se fuerza a la red a encontrar nuevos ‘caminos’, es decir, a emplear nuevas neuronas, que estaban siendo desaprovechadas, para resolver el problema.

- Capa de salida

Por último, como se desea clasificar en 5 tipos de clases y no 1 000, se añade una capa de salida que será de tipo *Dense*, que cuenta con tantas salidas como clases se pretendan clasificar, en este caso 5. Esta capa utiliza una función de activación de tipo **Softmax** o función exponencial normalizada:

Ecuación 17: Función exponencial normalizada

$$S(y_i) = \frac{e^{y_i}}{\sum_{k=1}^K e^{y_k}}$$

Donde:

$S$  es la salida de la función de activación de una neurona

$y_i$  es la puntuación obtenida por la red precedente de que pertenezca a la clase  $i$

$y_k$  son el resto de las puntuaciones de cada clase

$K$  es la cantidad de clases, en este caso 5

Por tanto, esta función recibe valores y devuelve porcentajes, entre 0 y 1, que definen la probabilidad de que la imagen que se está procesando pertenezca a un tipo de clase, esto es ideal para problemas de clasificación como el que se aborda.

Todo esto resulta en una red secuencial de 23 858 843 parámetros, de los cuales se entrenarán solamente 5 010 pertenecientes a la última capa de tipo *Dense*.

Model: "sequential"

| Layer (type)             | Output Shape | Param #  |
|--------------------------|--------------|----------|
| <hr/>                    |              |          |
| keras_layer (KerasLayer) | (None, 1001) | 23853833 |
| dropout (Dropout)        | (None, 1001) | 0        |
| dense (Dense)            | (None, 5)    | 5010     |
| <hr/>                    |              |          |
| Total params:            | 23,858,843   |          |
| Trainable params:        | 5,010        |          |
| Non-trainable params:    | 23,853,833   |          |

Figura 17: Arquitectura de la red construida

Fuente: Elaboración propia

#### 5.2.4. Compilación y entrenamiento

Conocido tanto el set de imágenes sobre el que se entrenará la red (epígrafe 5.1), como la estructura de esta (epígrafe 5.2) y, comprendiendo los diferentes tipos de optimizadores, así como la función de coste a emplear (epígrafe 4.2), se comenzó con el proceso de entrenamiento de la red neuronal.

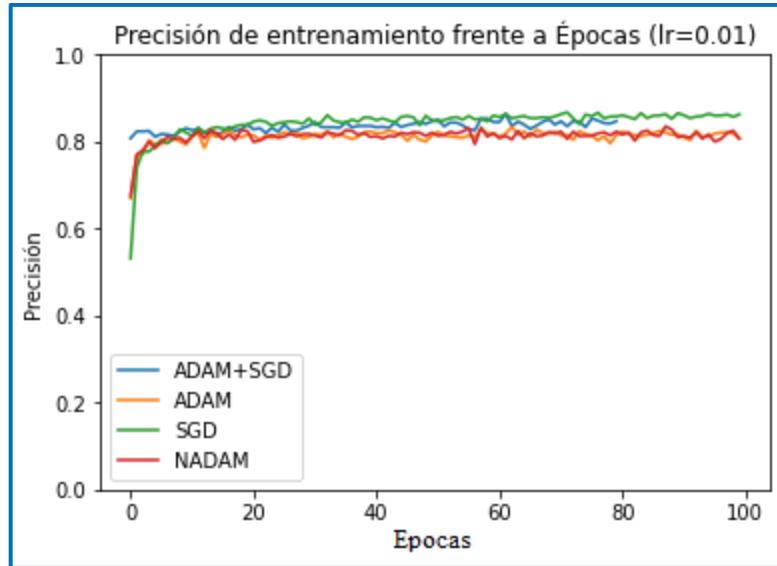
Como se explicó para la determinación de ciertos hiperparámetros u optimizadores, se necesita, en la mayoría de los casos, realizar múltiples experimentos y comparativas. Y es precisamente lo que se hizo para lograr obtener la mejor red dentro de unas determinadas características.

Se realizaron 5 experimentos, con 100 épocas cada uno; un *batch size* de 32, el recomendado para empezar; y una función de pérdida de tipo Entropía Cruzada Categórica.

Para los primeros 4 experimentos se empleó una tasa de aprendizaje de 0,01, la recomendada para comenzar normalmente los entrenamientos:

1. El primero de ellos consistía en una primera etapa de 20 épocas con un optimizador de tipo ADAM y 80 épocas con un optimizador de tipo SGD con un *Momentum* de 0,9.
2. En el segundo experimento se empleó un optimizador ADAM para las 100 épocas.
3. En el tercero se utilizó un SGD con un *Momentum* de 0,9.
4. En el cuarto se empleó el optimizador NADAM con una  $\beta_1$  de 0,9 y una  $\beta_2$  de 0,99.

En las siguientes gráficas se mostrarán las precisiones y las pérdidas obtenidas a lo largo de las 100 épocas de la etapa de entrenamiento y de la validación.

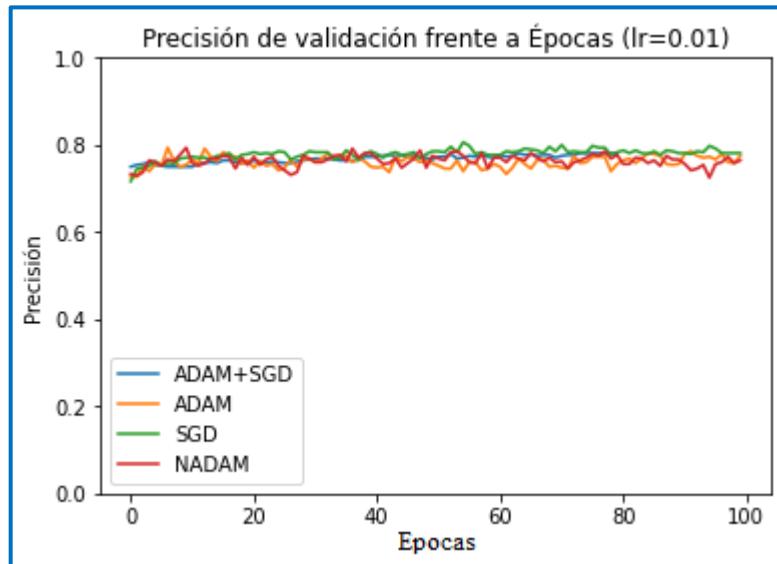


**Figura 18:** Precisión de entrenamiento frente a épocas de los optimizadores probados superpuestos

**Fuente:** Elaboración propia

Como se puede observar en la gráfica de la *Figura 18* el optimizador SGD es el que comienza con una precisión de entrenamiento más baja, entorno al 50%, pero asciende hasta superar el 80% y colocarse por encima del resto.

NADAM y ADAM responden prácticamente de la misma forma. En cuanto al optimizador ADAM+SGD<sup>7</sup> se queda ligeramente por debajo en cuanto a precisión del SGD.

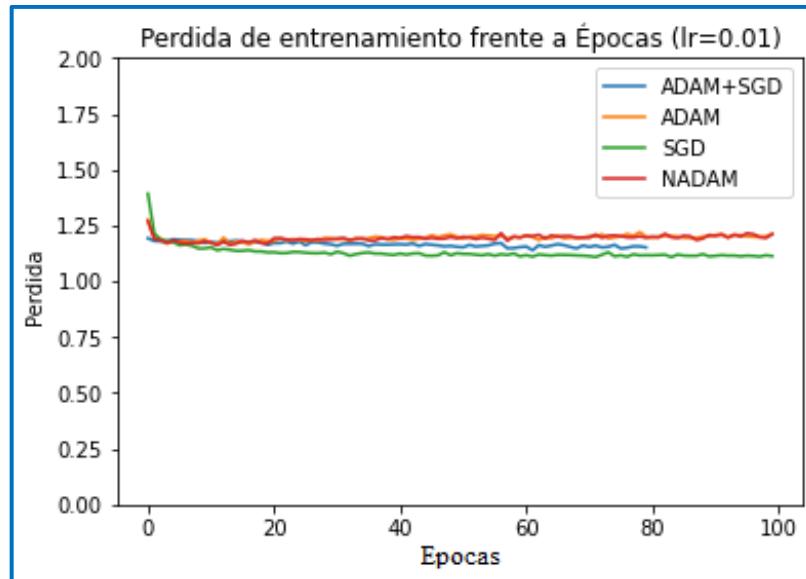


**Figura 19:** Precisión de validación frente a épocas de los optimizadores probados superpuestos

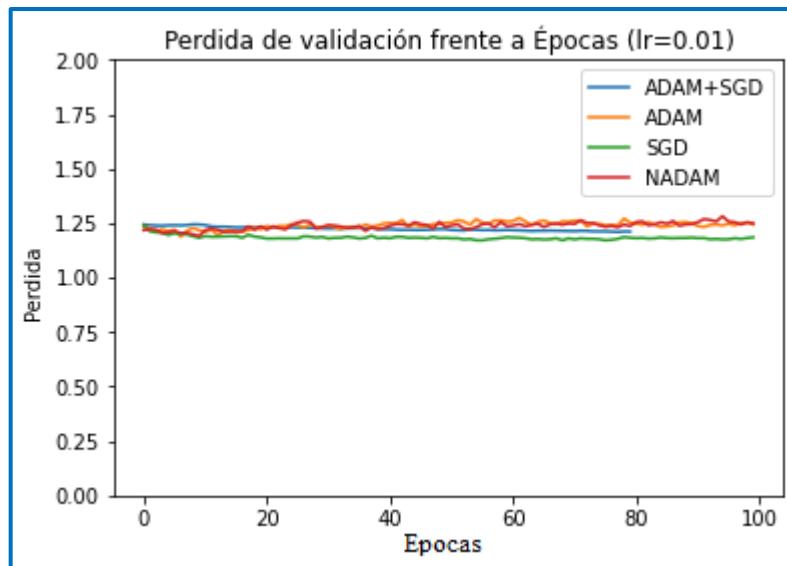
**Fuente:** Elaboración propia

Como la *Figura 19* demuestra, las 4 redes obtienen en las primeras épocas unas precisiones rondando el 75%. No obstante, el SGD se mantiene por encima en la mayor parte de los casos.

<sup>7</sup> Se están visualizando solo las 80 últimas épocas de la segunda etapa correspondiente al optimizador SGD.



**Figura 20:** Pérdida durante el entrenamiento de los optimizadores probados  
**Fuente:** Elaboración propia



**Figura 21:** Pérdida de validación de los optimizadores probados  
**Fuente:** Elaboración propia

En cuanto a las funciones de pérdida el SGD es el que tiene unos valores más bajos de su función de pérdida tanto en la fase de entrenamiento como en la de validación (*Figuras 20 y 21*).

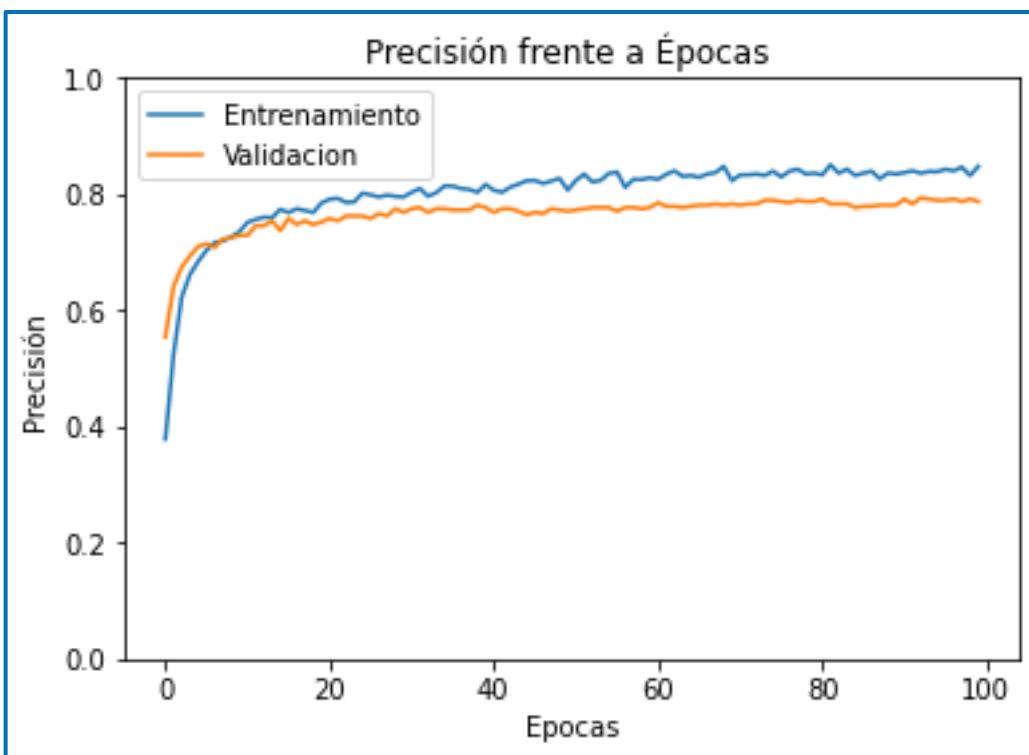
Maximos de precisión durante el entrenamiento alcanzados por:  
ADAM+SGD: 86.41%  
ADAM: 83.42%  
SGD: 86.76%  
NADAM: 83.52%  
Maximos de precisión durante la validación alcanzados por:  
ADAM+SGD: 78.12%  
ADAM: 79.37%  
SGD: 80.62%  
NADAM: 79.37%

**Figura 22:** Máximos de precisión de los optimizadores empleados

**Fuente:** Elaboración propia

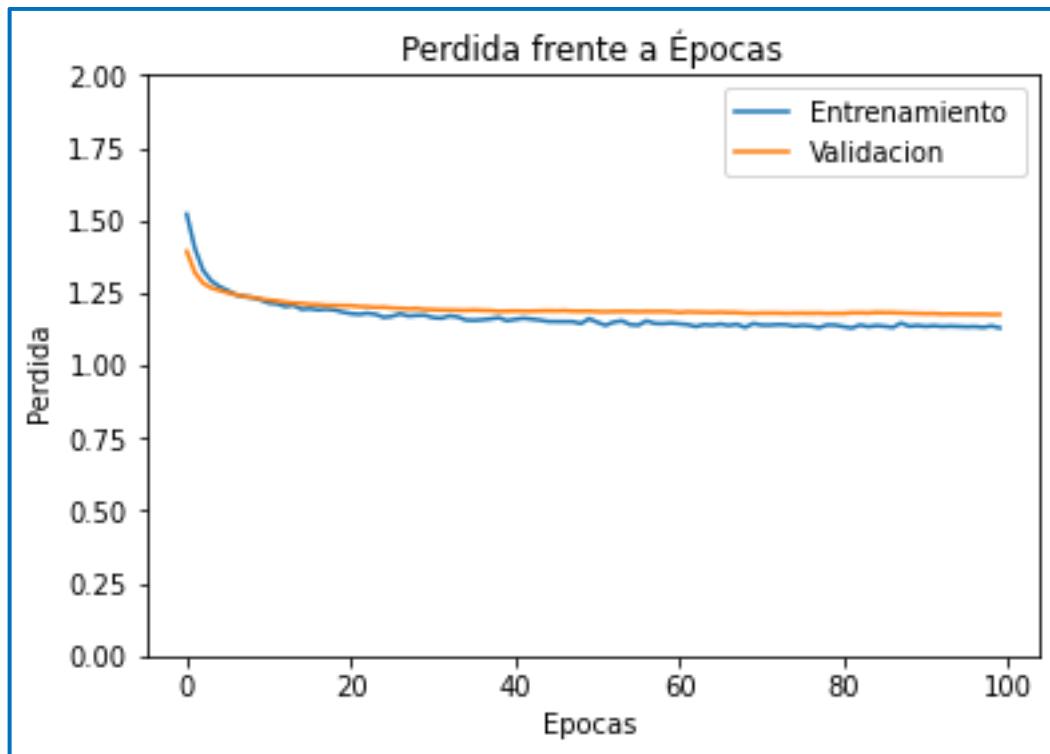
En la *Figura 22* se muestran los valores máximos de precisión obtenidos por cada uno de los optimizadores empleados. Como se puede comprobar el optimizador de tipo **SGD** es el que mejores resultados ha obtenido, con un 86.76% de aciertos durante la etapa de entrenamiento, y con un 80,62% durante la de validación.

Conociendo estos resultados, se optó por emplear la red SGD y se realizó un quinto experimento disminuyendo la tasa de aprendizaje a 0,001 para comprobar si se mejoraban los resultados.



**Figura 23:** Precisión del SGD con una tasa de aprendizaje del 0,001

**Fuente:** Elaboración propia



**Figura 24:** Pérdida del SGD con una tasa de aprendizaje del 0,001

**Fuente:** Elaboración propia

Como muestra la gráfica de las *Figuras 23* y *24*, no se mejoraron los resultados reduciendo la tasa de aprendizaje. De hecho, la precisión máxima obtenida esta vez fue inferior, del 85,07% durante el entrenamiento y del 79,37% durante la validación (*Figura 25*).

Maximos de precisión durante el entrenamiento :  
85.07%  
Maximos de precisión durante la validación:  
79.37%

**Figura 25:** Máximos de precisión del SGD con una tasa de aprendizaje del 0,001

**Fuente:** Elaboración propia

### 5.3. Test y resultados obtenidos

Para comprobar la eficacia de la red se sometió a un pequeño test sobre la base de imágenes. Es necesario recordar que para este proyecto no se contempló reservar una pequeña parte (normalmente del 10%) para realizar la etapa de test. Por lo que este experimento, aunque puede proporcionar una idea del estado de la red al final de su entrenamiento, no se puede comparar, ni mucho menos, con una etapa de test convencional en la que se pondría a prueba a la red con imágenes nuevas. Eso se realizará en la propia máquina depositando residuos sobre ella.

Esta pequeña prueba, por lo tanto, supuso clasificar 250 imágenes (50 de cada clase) y los resultados son los que se muestran en la *Figura 26*.

Resultados sobre 250 imágenes muestreadas:

|              |           |                  |
|--------------|-----------|------------------|
| Cartón: 44   | Fallos: 5 | Precisión: 88.0% |
| Metal: 45    | Fallos: 4 | Precisión: 90.0% |
| Papel: 49    | Fallos: 0 | Precisión: 98.0% |
| Plástico: 42 | Fallos: 7 | Precisión: 84.0% |
| Vidrio: 46   | Fallos: 3 | Precisión: 92.0% |

**Figura 26:** Resultados del test sobre la red elegida

**Fuente:** Elaboración propia

Es comprensible que la clase que peores resultados obtenga sea la del plástico porque se suelen presentar formas y colores mucho más variados que con otros materiales.

**Tabla 1:** Resultados obtenidos de los experimentos realizados con diferentes optimizadores

**Fuente:** Elaboración Propia

| N.º de experimento | Tasa de aprendizaje | Optimizador | Precisión (%) |               |
|--------------------|---------------------|-------------|---------------|---------------|
|                    |                     |             | Validación    | Entrenamiento |
| 1                  | 0,01                | ADAM+SGD    | 78,12         | 86,41         |
| 2                  |                     | ADAM        | 79,37         | 83,42         |
| 3                  |                     | SGD         | <b>80,62</b>  | <b>86,76</b>  |
| 4                  |                     | NADAM       | 79,37         | 83,52         |
| 5                  | 0,001               | SGD         | 79,37         | 85,07         |

#### 5.4. Congelación del modelo e inferencia

Una vez entrenada la red es necesario guardarla o congelarla (*freeze*) para poder comenzar a introducir imágenes y que las clasifique, es decir, para poder realizar sobre ella un proceso de inferencia.

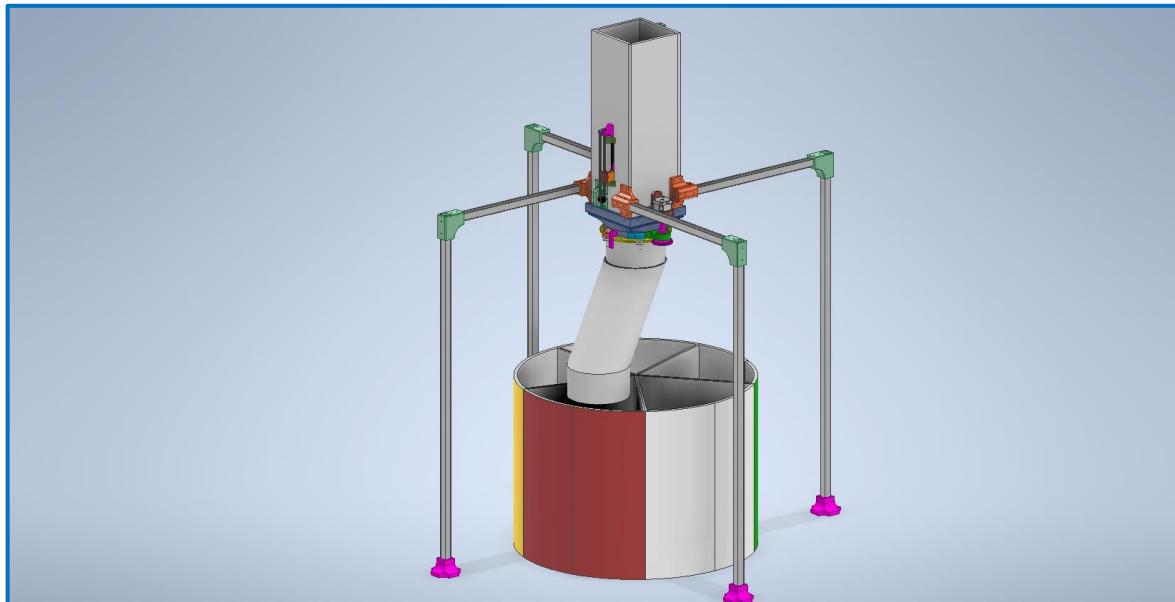
Para guardar la red en *Tensorflow*, como se observa en el código adjuntado, se puede guardar en formato PB. Pero existe un formato mucho más liviano ya que fue diseñado para desarrollar redes neuronales que fuesen empleadas en dispositivos móviles, denominado TFLITE. Será por tanto este formato el que se elija para guardar el modelo de la red e introducirlo en el dispositivo de la máquina que se encargará de la inferencia, en este caso una Jetson Nano de la marca Nvidia®.

Para realizar la inferencia se ha desarrollado un *script* clasificador.py<sup>8</sup>. Tiene un método “clasificar” al cual se le pasa un “interpretador” por parámetro y la ruta de la imagen que debe ser procesada. El interpretador no es más que un método de la librería “tf.lite” que lee el modelo guardado en formato TFLITE en un directorio y lo interpreta. El proceso de carga del modelo se hará desde el programa que gestiona el funcionamiento de la máquina (principal.py), pues el proceso de carga del modelo se demora un par de segundos en el tiempo y es conveniente

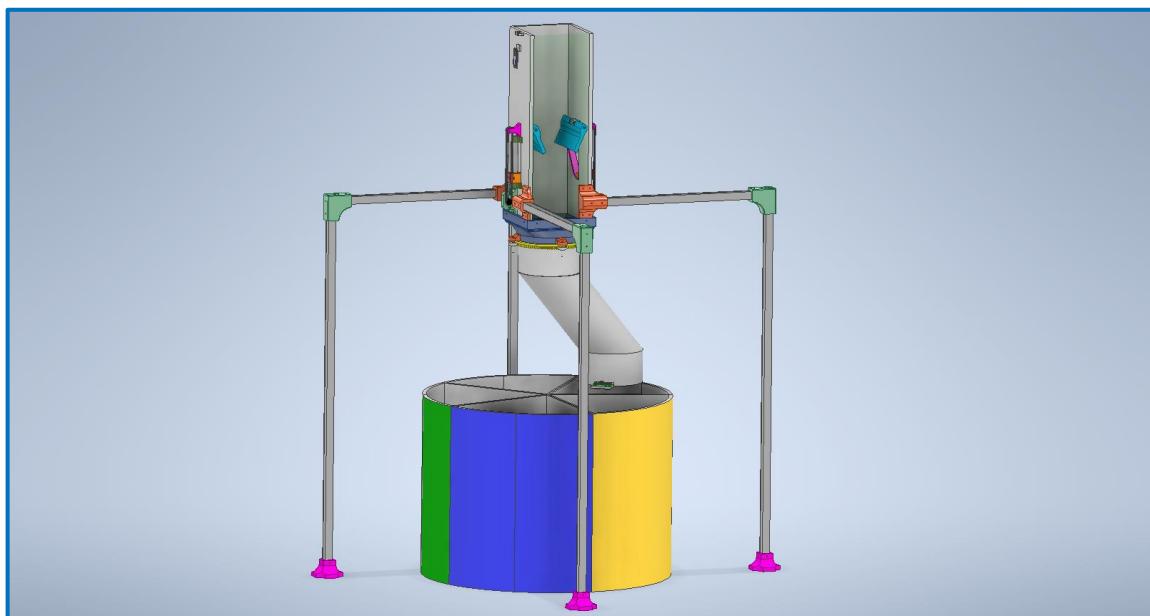
<sup>8</sup> Código completo y comentado en el Anexo 15.2.3 Código del Jetson Nano

hacerlo una sola vez al principio de la ejecución. La función “clasificar” por tanto devuelve en una *string* el material que la red considera que pertenece la imagen proporcionada, excepto si la precisión del resultado es menor a un 60%, en ese caso se considerará que el residuo fotografiado pertenece a la clase “orgánico”.

## 6. DISEÑO DEL PROTOTIPO



**Figura 27:** Vista general del modelo 3D de la máquina  
**Fuente:** Elaboración propia



**Figura 28:** Vista general del interior de la máquina  
**Fuente:** Elaboración propia

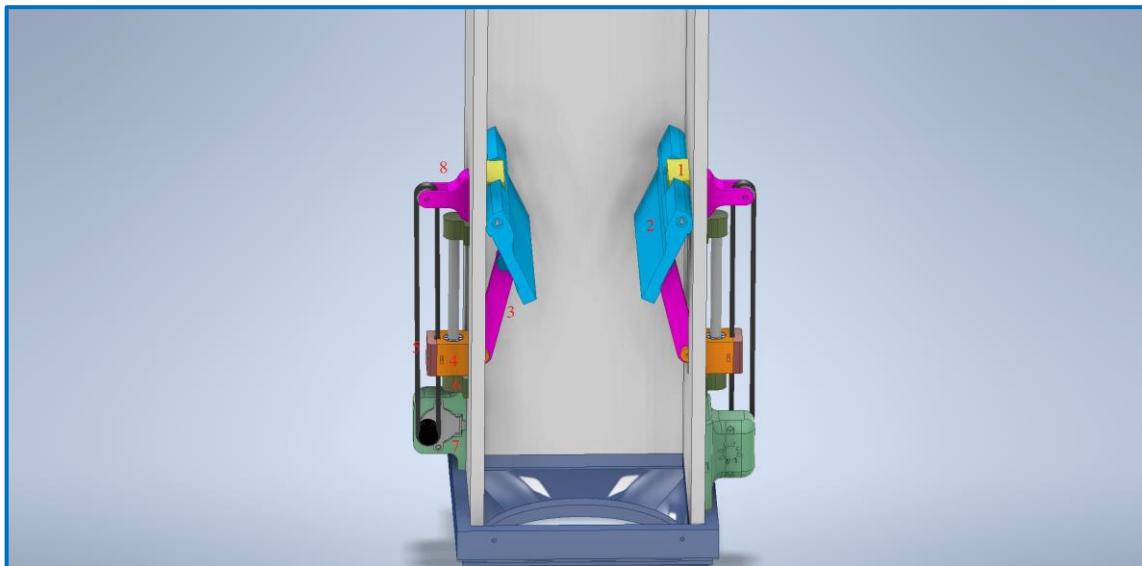
La máquina se diseñó con el propósito de cumplir las necesidades de movimiento para gestionar los residuos. Se compone de **26** tipos de piezas diseñadas mediante el software Autodesk Inventor, por la familiaridad con la que contaba el autor de este trabajo.

Las piezas, una vez diseñadas, fueron exportadas a formato de litografía (STL) para así poder ser impresas mediante una impresora 3D de tipo FDM. Es necesario tener en cuenta este aspecto, pues la mayoría de las piezas han sido impresas con un diámetro del nozzle de entre 0,2 y 0,5 mm, a excepción de la pieza más voluminosa que se imprimió con boquilla de 1 mm para acelerar el proceso. La altura de capa utilizada es de 0,2 para las piezas pequeñas y precisas y 0,3 para las más grandes. Con todo esto las medidas de las piezas se han calculado atendiendo

a estos parámetros para que sean replicables durante la impresión. Como ejemplo, el caso de un eslabón, que se comentará con más detalle más adelante, en el que la distancia entre el radio exterior y el interior debería de ser una distancia que, al dividirla por el diámetro del nozzle, resultase en un número entero de líneas de impresión. También, a lo largo del diseño de las piezas, se ha pretendido reducir al máximo los tiempos de impresión, ya sea diseñando las piezas lo más reducidas y optimizadas posibles, segmentándolas, o evitando voladizos que requiriesen soporte durante la impresión.

## 6.1. Mecanismo compuertas

A continuación, se detallará el mecanismo implementado para la apertura y cierre de las compuertas. En este espacio será donde comenzará el proceso de clasificación de un residuo, cuando se deposite sobre las compuertas cerradas.



**Figura 29:** Vista general del mecanismo de compuertas

**Fuente:** Elaboración propia

Consiste en un mecanismo de biela-manivela, en el que una sujeción (en la imagen representado por el número 1) fija a la pared de la máquina la compuerta que hace la función de manivela (2) y la biela, un eslabón (3) que se fija por un extremo a la compuerta y por el contrario a una pieza que hace de émbolo (4), para ello se emplea un rodamiento lineal IGU® RJ4JP 01 08 alojado en dicha pieza para así poder efectuar un deslizamiento lineal, de este rodamiento se elimina el movimiento de rotación al trasladarse esta pieza entre las aberturas de las paredes de la máquina. No obstante, aunque cumple su función, sería más recomendable emplear guías lineales para así restringir el movimiento a traslación de tipo corredera y no cilíndrica, pero se optó por los IGU por tres motivos principales: son económicamente más rentables; el autor de este proyecto ya disponía de ellos; y, principalmente, porque cumplen su función a la perfección.

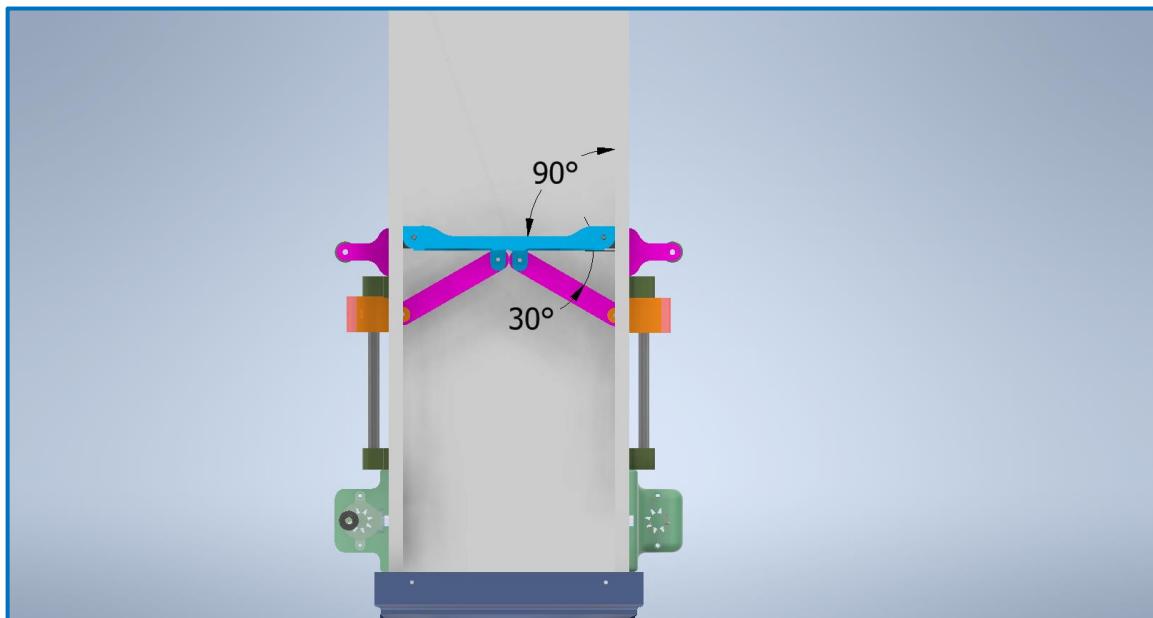
Por tanto, dicha pieza se trasladará a lo largo de una barra cilíndrica de aluminio fijada a las paredes de la maquina mediante dos sujetaciones (6). Para ejecutar el movimiento lineal se empleará un motor “paso a paso 28BYJ-48” fijado mediante una pieza especialmente diseñada

(7) a las paredes. El motor desplazará una cremallera GT2 fijada al émbolo, con una pieza (5) que en su interior recrea el perfil de la cremallera para así fijar la cremallera eficazmente al émbolo. En el extremo opuesto al motor se encuentra una polea para GT2 de 16 dientes colocada sobre una porta-poleas (8) para así completar el recorrido de dicha cremallera.

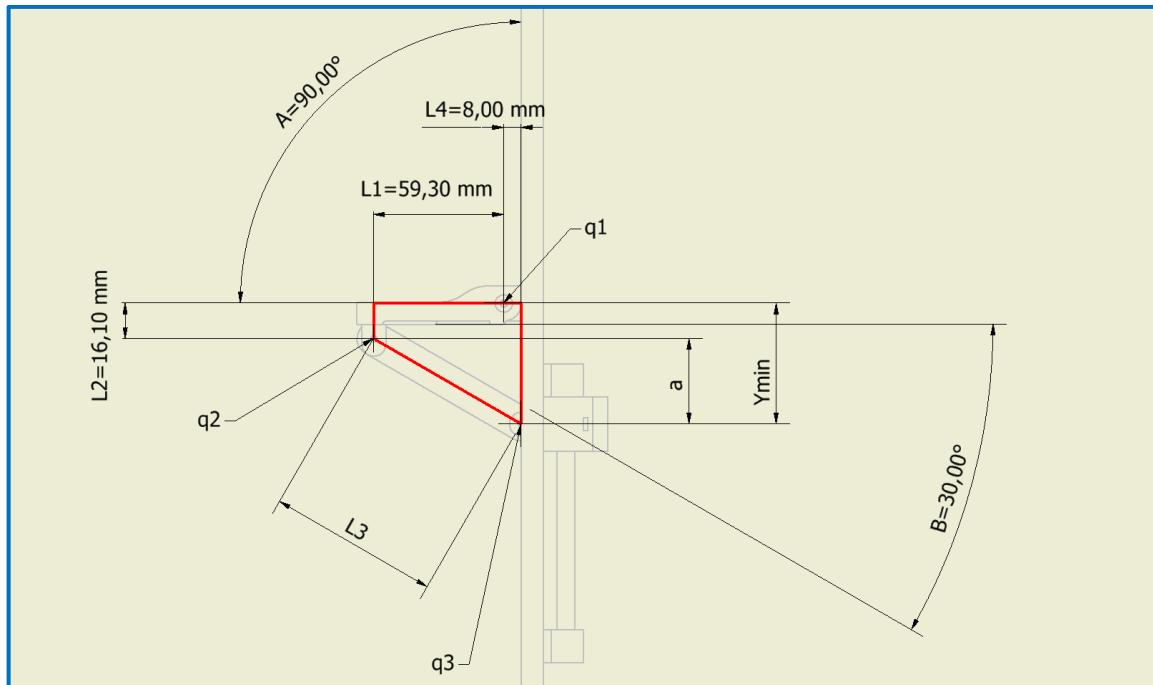
Para diseñar este mecanismo se tenía como objetivo primordial que la apertura entre las compuertas fuese lo suficientemente amplia como para permitir que un residuo pequeño pudiese acceder sin dificultades, por ejemplo, una botella de plástico pequeña estándar de unos 6 cm de diámetro y 15 cm de alto.

Para alcanzar este objetivo se deben tener en cuenta 3 longitudes: la longitud de la manivela, es decir la compuerta; la longitud de la biela; y el rango de movimiento del émbolo.

Para reducir al máximo el tamaño de la máquina, pero permitiendo que fuese posible procesar residuos pequeños se decidió que el ancho de las paredes interiores de la máquina fuese de 150 mm, es decir, que cada compuerta tuviese un tamaño de 150 x 75 mm. Por otro lado, para facilitar el cierre se consideró que un ángulo de 30° entre la biela y la compuerta era razonable cuando esta estuviese cerrada, es decir, formando 90° con la superficie de la pared.



**Figura 30:** Mecanismo de compuertas. Compuertas cerradas  
**Fuente:** Elaboración propia



**Figura 31:** Mecanismo de compuertas. Cálculos geométricos de diseño de compuertas cerradas

**Fuente:** Elaboración propia

Teniendo en cuenta los parámetros indicados y conociendo el diseño de la compuerta, que posteriormente se justificará, tal y como se representa en la *Figura 31*, se aprecia que la distancia horizontal entre el eje  $q_1$  y la pared es  $L_4 = 8 \text{ mm}$ ; la distancia horizontal entre el eje  $q_1$  y el eje  $q_2$  es  $L_1 = 59,3 \text{ mm}$ ; y la distancia vertical es  $L_2 = 16,1 \text{ mm}$ . Se puede entonces resolver la distancia vertical  $Y_{min}$  que separa el eje  $q_1$  del eje  $q_3$ . Esta distancia será necesaria para colocar correctamente la compuerta respecto a la parte superior de la hendidura por la que se desplaza el émbolo.

En el esquema de la imagen superior se deduce la siguiente ecuación:

**Ecuación 18: Cálculo de la distancia entre el eje de la compuerta cerrada ( $q_1$ ) y el eje del émbolo ( $q_3$ )**

$$Y_{min} = L_2 + a$$

Siendo  $a$ :

**Ecuación 19: Distancia vertical entre eje  $q_2$  y eje  $q_3$  (compuertas cerradas)**

$$a = (L_1 + L_4) * \tan(B)$$

Con todo esto

$$Y_{min} = 54,95 \text{ mm}$$

Para calcular la distancia entre ejes del eslabón o biela se aplicó la siguiente fórmula trigonométrica:

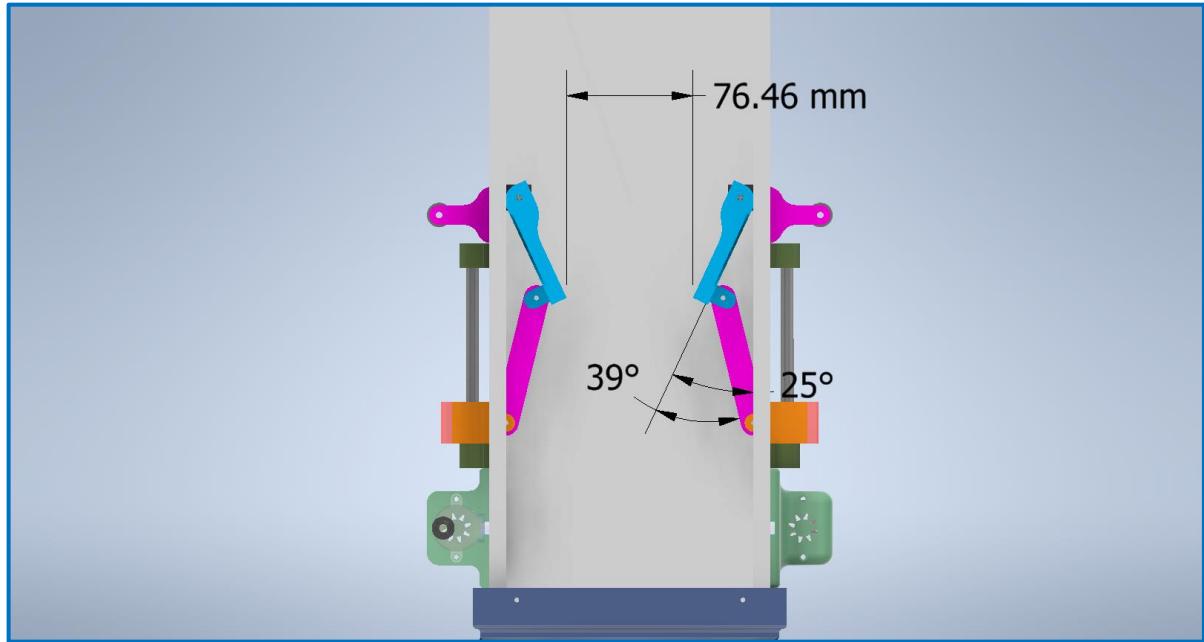
**Ecuación 20: Cálculo de la longitud del eslabón**

$$L_3 = \frac{L_1 + L_4}{\cos(B)}$$

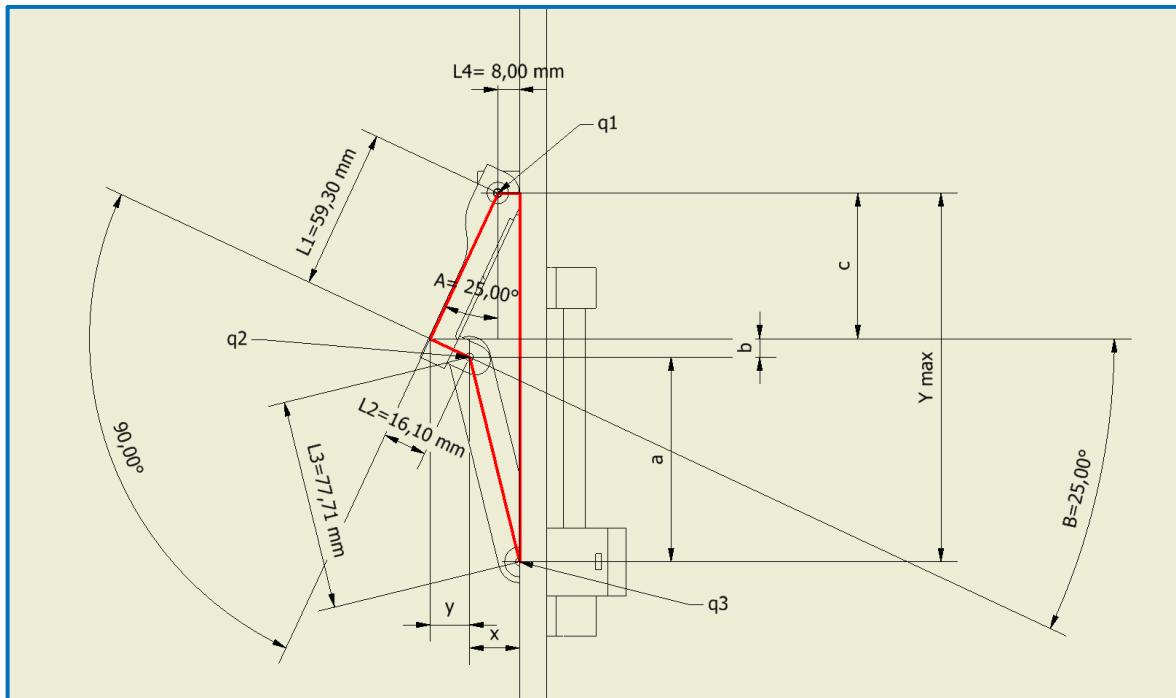
Con lo que

$$L_3 = 77,71 \text{ mm}$$

Por otro lado, se estableció que las compuertas al abrirse formasen un ángulo de 25° con la pared pues con esto formarían una distancia entre las dos compuertas de 76,46 mm, suficiente para que un residuo pequeño pudiese transitar con total facilidad.



**Figura 32:** Mecanismo de compuertas. Compuertas abiertas  
**Fuente:** Elaboración propia



**Figura 33:** Mecanismo de compuertas. Cálculos geométricos de diseño con compuertas abiertas  
**Fuente:** Elaboración propia

Conociendo ya  $L_1$ ,  $L_2$ ,  $L_3$  y  $L_4$ , se puede calcular la distancia entre el eje  $q_1$  y  $q_3$  que habrá cuando las compuertas estén totalmente abiertas.

Del esquema anterior se deduce que:

**Ecuación 21: Cálculo de la distancia entre el eje de la compuerta abierta ( $q_1$ ) y el eje del émbolo ( $q_3$ )**

$$Y_{max} = a + b + c$$

Siendo:

**Ecuación 22: Cálculo de componente c**

$$c = L_1 * \cos(A)$$

**Ecuación 23: Cálculo de componente b**

$$b = L_2 * \sin(B)$$

**Ecuación 24: Cálculo de componente a**

$$a = \sqrt{L_3^2 - x^2}$$

El ángulo A es conocido, es el que forma la compuerta con la pared, y es de  $25^\circ$ . Por tanto, el ángulo B también es de  $25^\circ$ . En cuanto a la longitud x:

**Ecuación 25: Cálculo de componente x**

$$x = (L_1 * \sin(A) + L_4) - y$$

Siendo y:

**Ecuación 26: Cálculo de componente y**

$$y = L_2 * \cos(B)$$

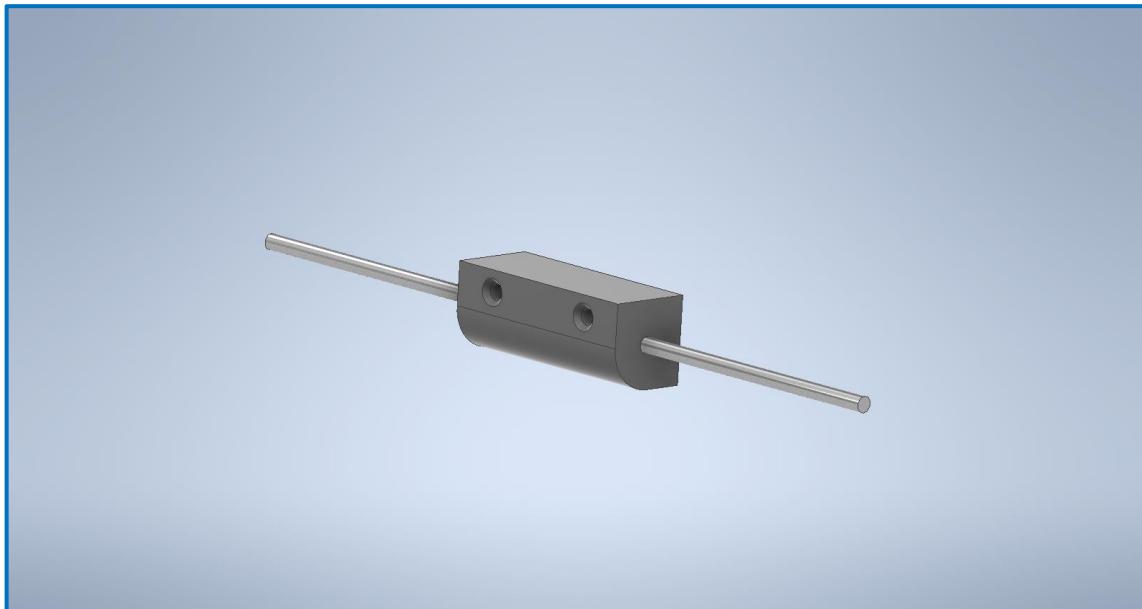
Con todo esto

$$Y_{max} = 136.03 \text{ mm.}$$

A continuación, se detalla cada pieza diseñada para esta tarea, acompañando de imágenes de las mis

mas:

### 6.1.1. Sujeción compuerta<sup>9</sup>



**Figura 34:** Sujeción de compuerta

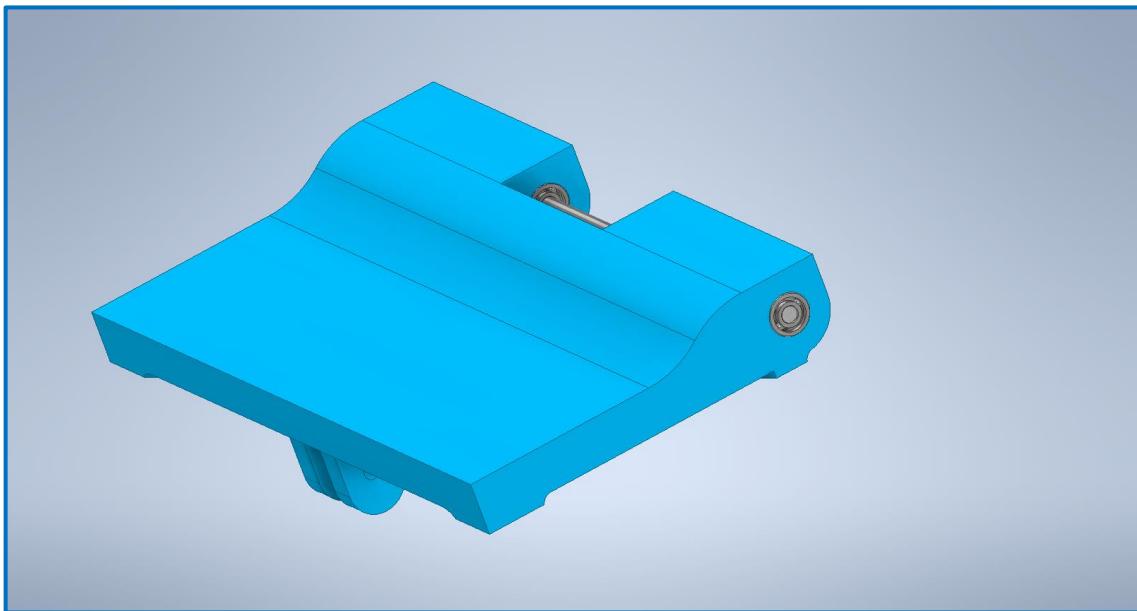
**Fuente:** Elaboración propia

Esta pieza permite fijar las compuertas a las paredes de la máquina mediante dos tornillos de 3 mm de diámetro, e introducir una varilla de acero de 3 mm de diámetro y 150 mm de largo sobre la que rotará la compuerta.

En la parte inferior cuenta con un empalme de radio 8 mm que permite que la compuerta descienda sin provocar ningún tipo de interferencia con esta pieza.

<sup>9</sup> Plano en el Anexo 15.1 Planos. Plano 1: Sujeción compuerta

### 6.1.2. Compuerta (*manivela*)<sup>10</sup>



**Figura 35:** Compuerta. Vista superior delantera

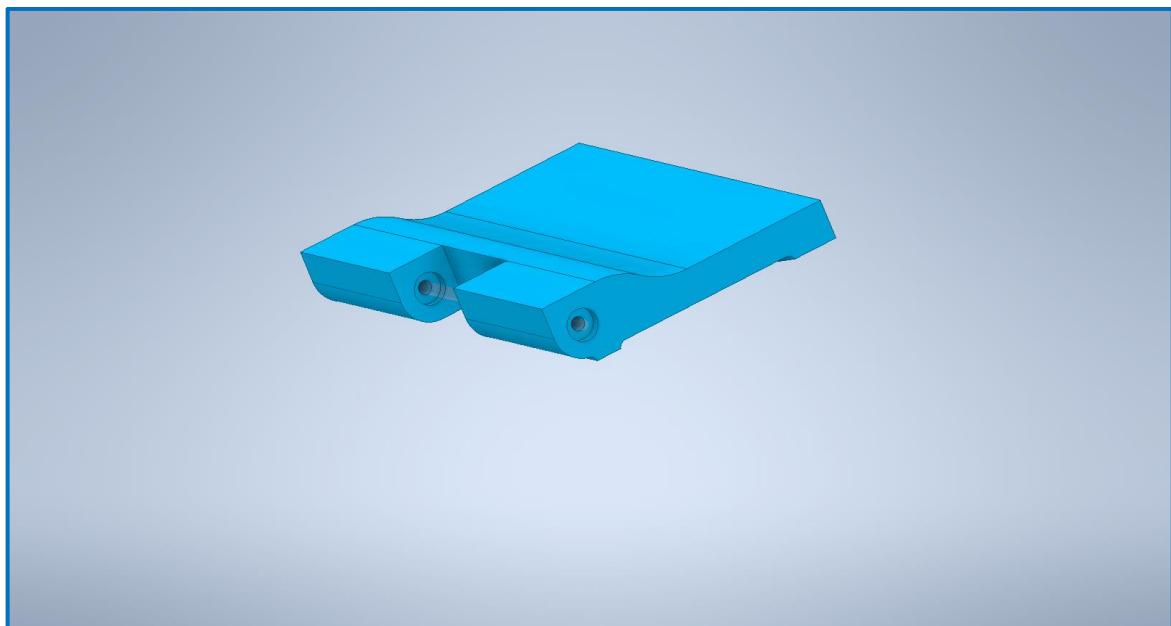
**Fuente:** Elaboración propia

La compuerta consiste en una plataforma de 150 mm de ancho y 75 mm de largo.

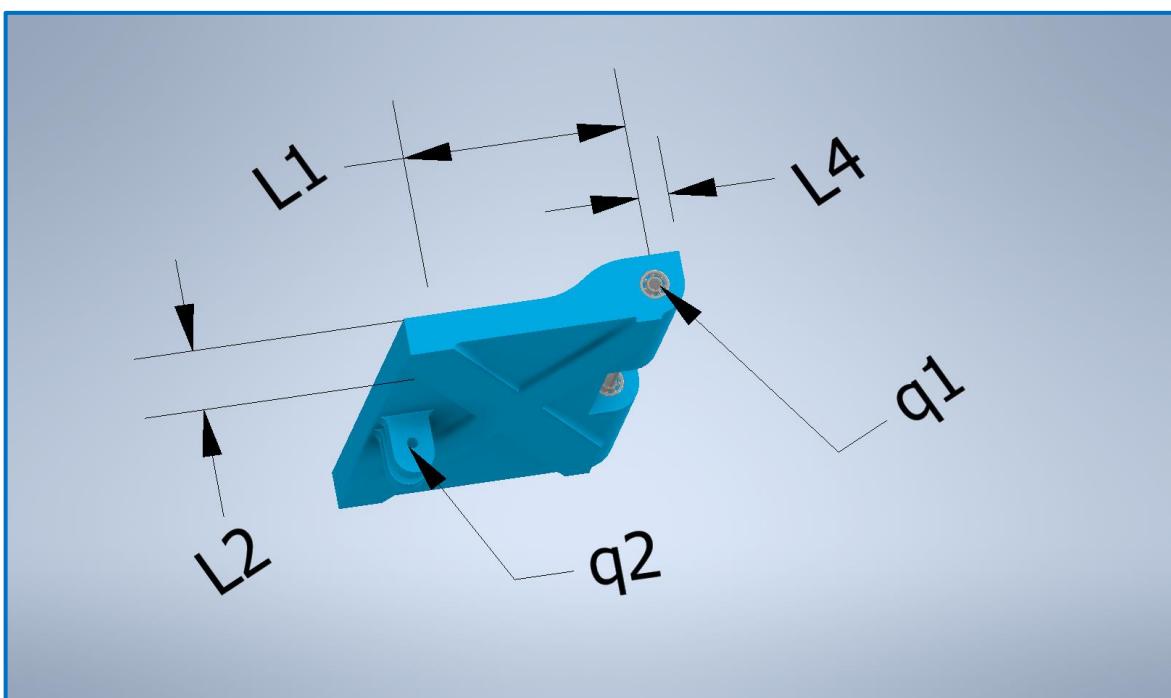
El perfil es curvado para que los residuos se concentren en el centro y facilitar así su caída, reducir el grosor y, por tanto, tiempo de impresión y permitir tener un espacio en el extremo opuesto donde alojar los rodamientos y la varilla (eje  $q_1$ ).

Cuenta con un orificio de 3,4 mm que lo atraviesa a lo ancho para que pueda introducirse de forma holgada la varilla de 3 mm. Esta varilla rotará respecto a la compuerta gracias a 4 rodamientos MR83ZZ (diámetro exterior 8 mm, diámetro interior 3 mm, espesor 3 mm) situados en los orificios de diámetro 8 mm y profundidad 3 mm (*Figura 36*).

<sup>10</sup> Plano en el Anexo 15.1 Planos. Plano 2: Compuerta



**Figura 36:** Compuerta. Vista superior trasera  
**Fuente:** Elaboración propia



**Figura 37:** Compuerta. Esquema de referencia para distancias  
**Fuente:** Elaboración propia

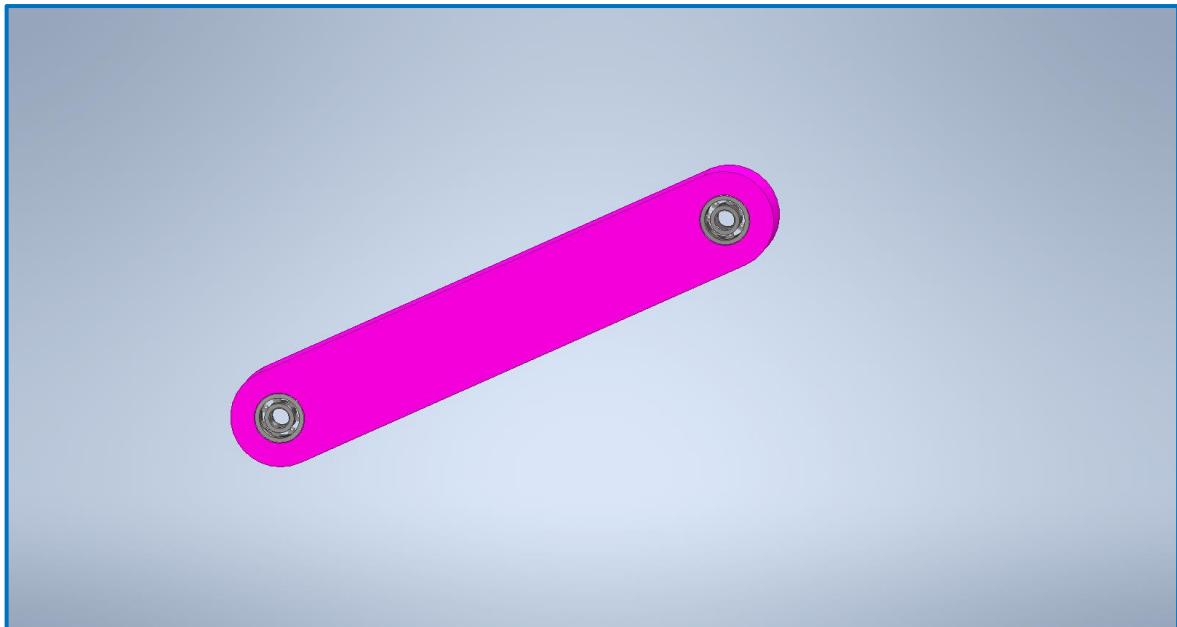
En la parte inferior se refuerza la estructura con un perfil en forma de X.

La distancia entre el eje  $q_1$  y la pared ( $L_4$ ) es de 8 mm pues así permite que, siendo el rodamiento de 4 mm de radio, dejar otros 4 mm de contorno para garantizar una buena solidez de la impresión. En cuanto a la distancia  $L_1$  entre los ejes  $q_1$  y  $q_2$  es el resultado de colocar el eje  $q_2$  en el extremo contrario de la compuerta para maximizar el potencial ángulo de giro de la compuerta y reducir la distancia que el émbolo debe de realizar, teniendo en cuenta, por supuesto, paredes de grosor 4 mm.

La distancia entre el eje  $q_2$  y la compuerta  $L_2$  es, además, el necesario para que el eslabón no interfiera cuando está rotando sobre este eje.

Los empalmes de la compuerta por el lado del eje  $q_1$  se diseñaron con un radio de 8 mm para permitir una rotación máxima de la compuerta de  $25^\circ$  (como se había establecido) sin provocar ninguna interferencia con la pared.

### 6.1.3. Eslabón (*biela*)<sup>11</sup>

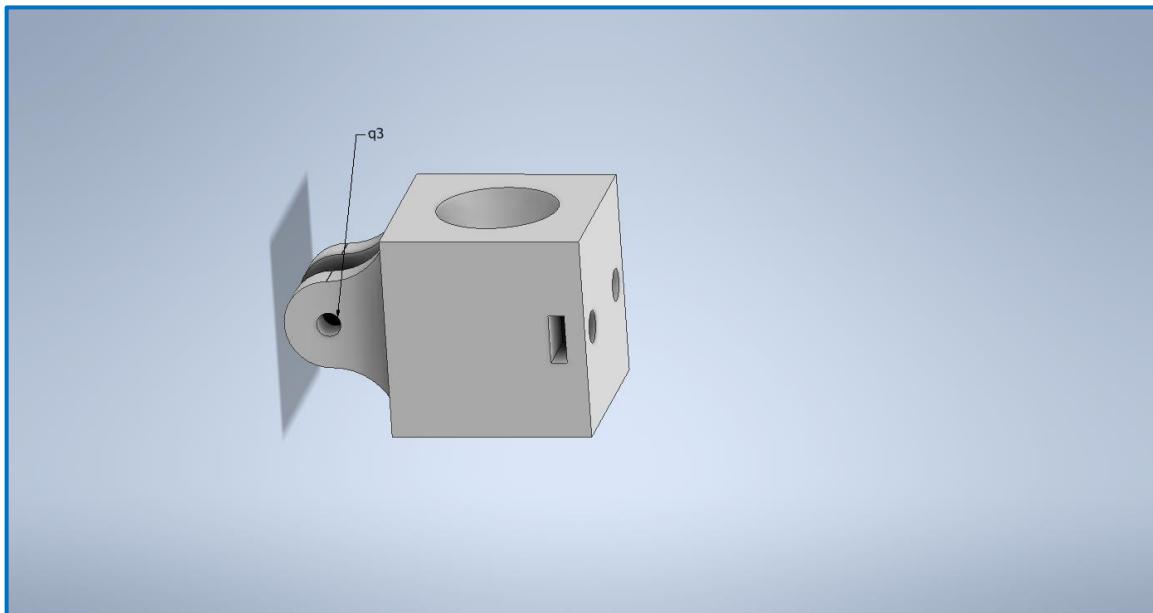


**Figura 38:** Eslabón  
**Fuente:** Elaboración propia

El eslabón es una pieza que cuenta con dos orificios (eje  $q_2$  y  $q_3$ ) en los que introducir dos rodamientos MR83ZZ para facilitar la rotación respecto a la compuerta y el émbolo, estos ejes se colocaron a una distancia  $L_3$  que ha sido calculada previamente al comienzo de este epígrafe. En cuanto al diámetro exterior se decidió de forma que, a la hora de imprimirse, diese como resultado un número entero de líneas de impresión y garantizase robustez a la pieza.

<sup>11</sup> Plano en el Anexo 15.1 Planos. Plano 3: Eslabón

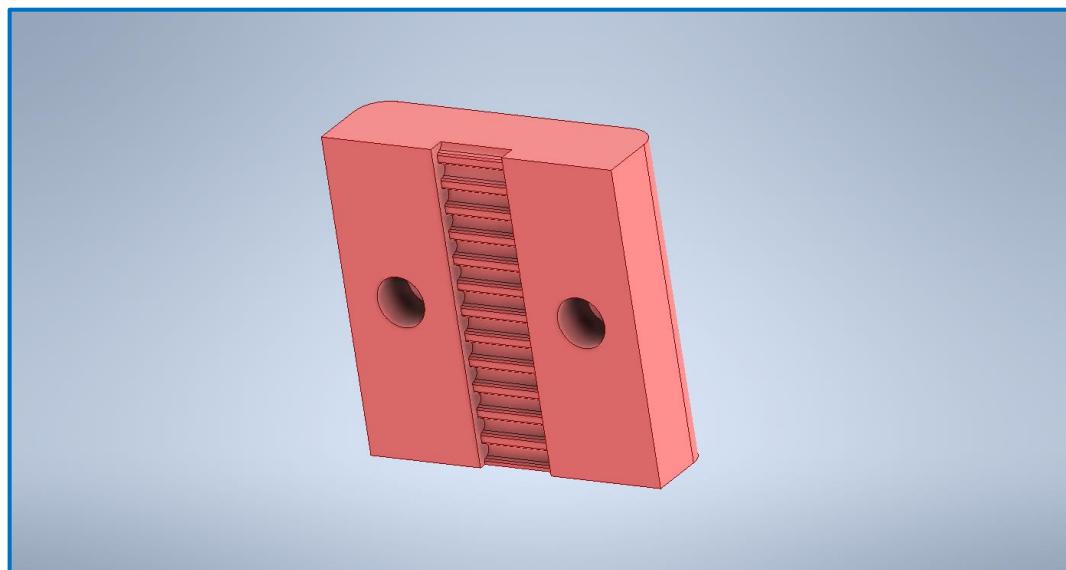
#### 6.1.4. Émbolo<sup>12</sup>



**Figura 39:** Émbolo  
**Fuente:** Elaboración propia

El émbolo cuenta con una zona en la que se encontrará el eje  $q_3$  que conecta el eslabón y el embolo, un orificio de diámetro 15 mm para insertar un rodamiento lineal IGU® RJ4JP 01 08, y dos orificios de diámetro 4 y otros dos orificios perpendiculares de perfil cuadrado para insertar los tornillos y sus correspondientes tuercas que fijarán la pieza que engancha la correa con el émbolo.

#### 6.1.5. Sujeción Correa<sup>13</sup>



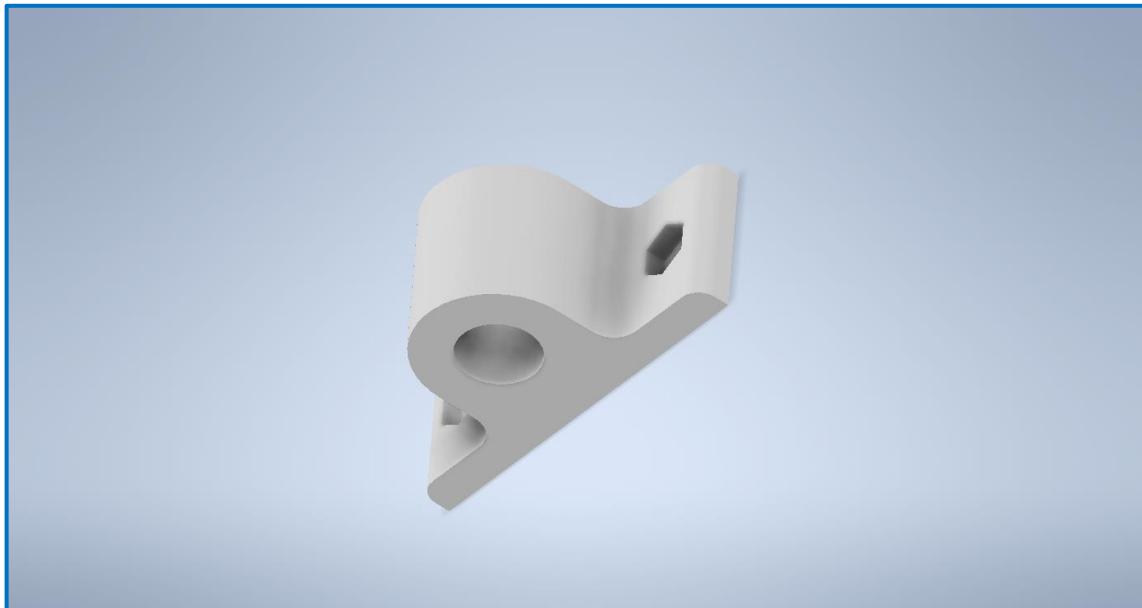
**Figura 40:** Sujeción correa  
**Fuente:** Elaboración propia

<sup>12</sup> Plano en el Anexo 15.1 Planos. Plano 4: Émbolo

<sup>13</sup> Plano en el Anexo 15.1 Planos. Plano 5: Sujeción correa

Esta pieza cuenta con un perfil igual a la correa GT2 para así engancharla eficazmente. También cuenta con dos orificios para atornillar la pieza al émbolo.

#### 6.1.6. Sujeción varillas<sup>14</sup>



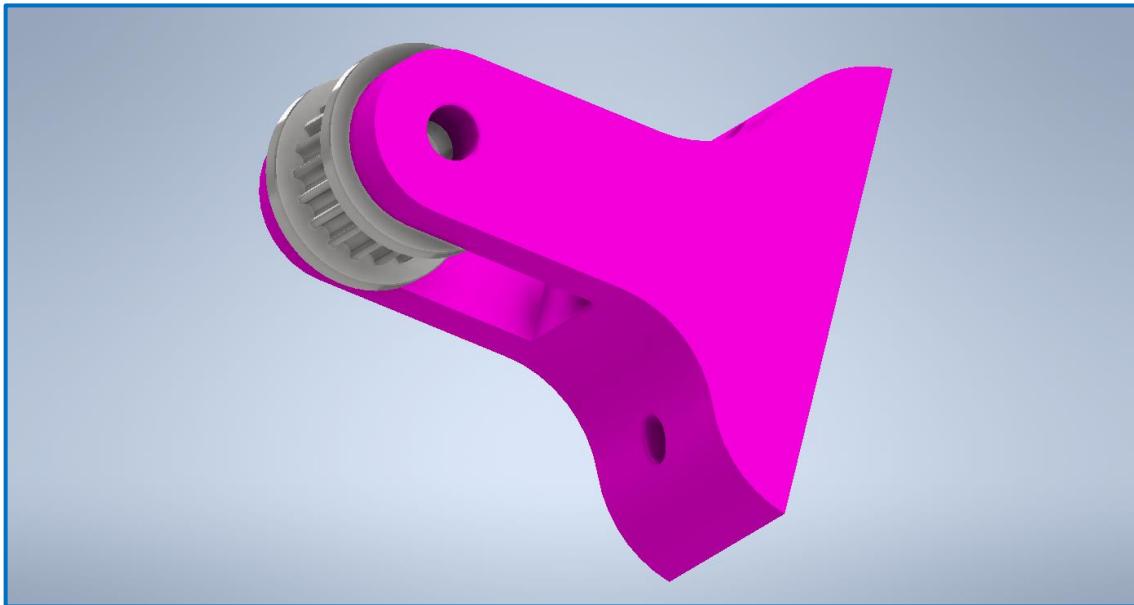
**Figura 41:** Sujeción varillas

**Fuente:** Elaboración propia

Esta pieza se encarga de fijar la varilla de 8 mm sobre la que se desplaza el émbolo a la pared de la máquina. Para ello cuenta con un agujero de diámetro 8 mm y dos orificios perpendiculares a la superficie de la pared para atornillarla y fijarla con tuercas de perfil hexagonal que se asentarán en la pieza gracias a dos huecos con este tipo de perfil.

<sup>14</sup> Plano en el Anexo 15.1 Planos. Plano 6: Sujeción varillas

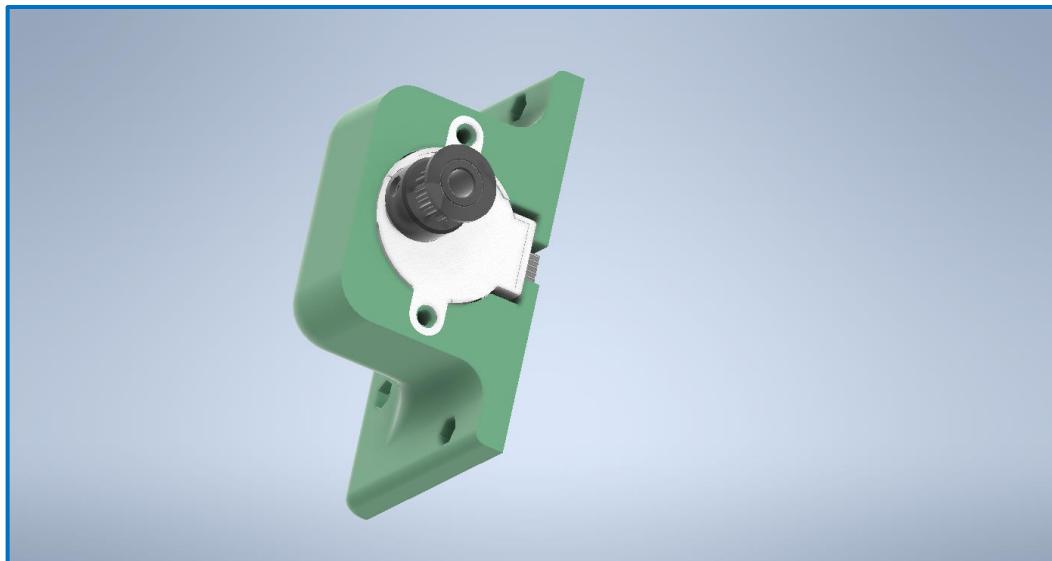
### 6.1.7. Porta poleas<sup>15</sup>



**Figura 42:** Porta poleas  
**Fuente:** Elaboración propia

En la porta-poleas se asentará una polea de 16 dientes para la correa GT2. La altura a la que se encuentran los orificios, es decir, sobre la que rotará la polea, se ha calculado en función de la altura que tendrá el eje del rotor del motor que moverá dicha cadena y la altura a la que estará el enganche de la cadena del émbolo respecto a la pared. Con esto se facilitará el movimiento de la correa y por tanto del émbolo. En la parte inferior de esta pieza se encuentran dos orificios para fijarla a la pared de la máquina.

### 6.1.8. Sujeción motor compuertas<sup>16</sup>

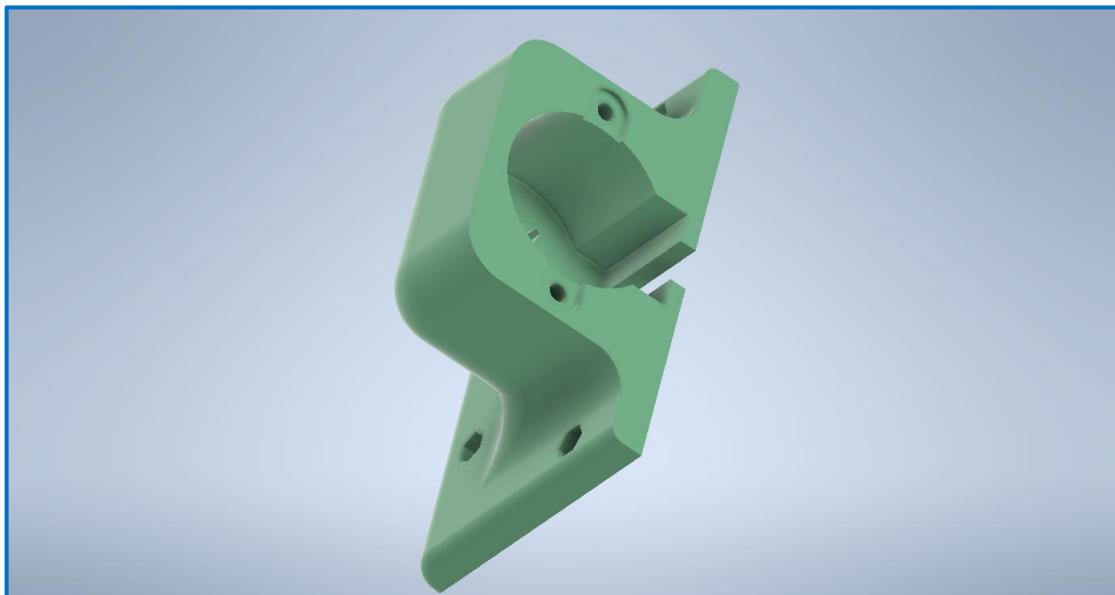


**Figura 43:** Sujeción del motor de la compuerta. Motor 28BYJ-48 insertado  
**Fuente:** Elaboración propia

<sup>15</sup> Plano en el Anexo 15.1 Planos. Plano 7: Porta-poleas

<sup>16</sup> Plano en el Anexo 15.1 Planos. Plano 8: Sujeción motor

La sujeción del motor permite fijarlo a la pared y evitar así cualquier movimiento que perturbe la ejecución de la subida o bajada de las compuertas o del selector, pues esta pieza como se verá en el siguiente epígrafe, también se emplea en este mecanismo.

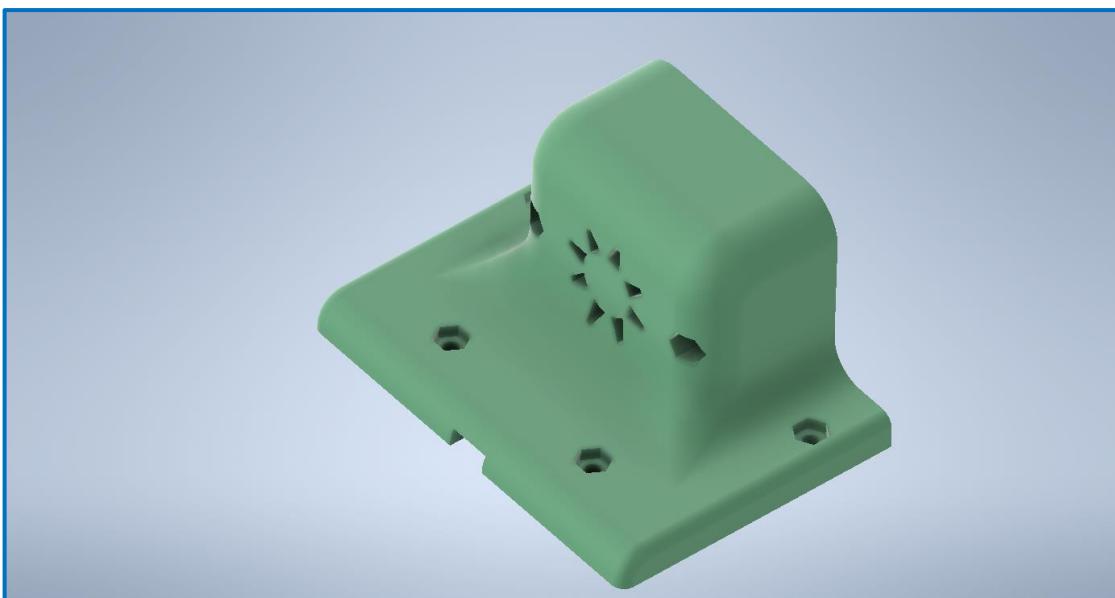


**Figura 44:** Sujeción del motor de la compuerta. Vista delantera

**Fuente:** Elaboración propia

Tiene un orificio central con la forma del motor para insertarlo, y dos orificios de 3 mm de diámetro para introducir tornillos que lo sujeten.

En la parte inferior hay una ranura por la que atraviesan la pieza los cables de dicho motor.



**Figura 45:** Sujeción del motor de la compuerta. Vista trasera

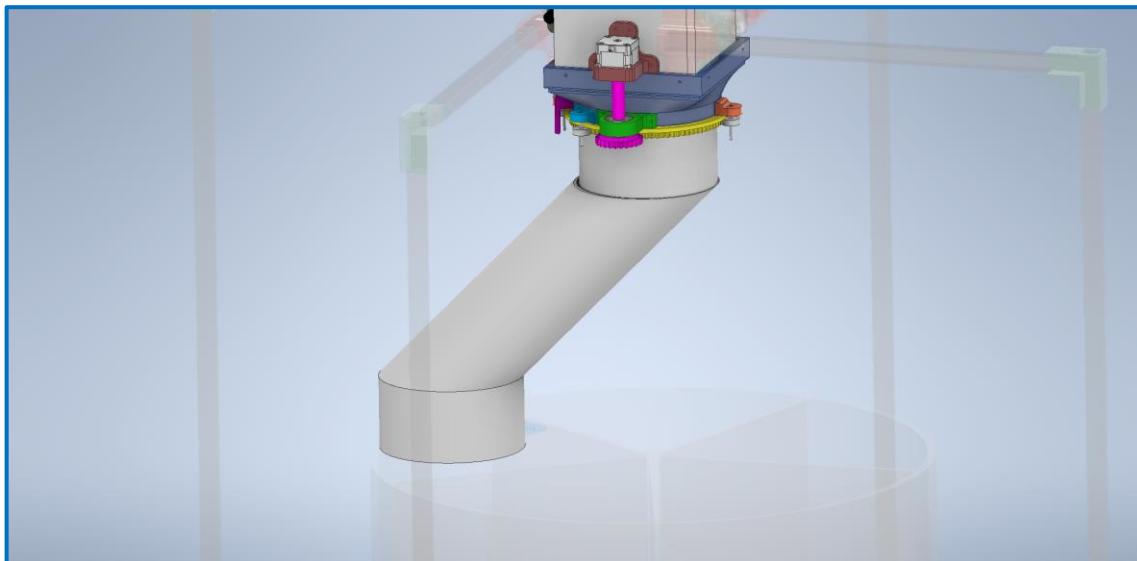
**Fuente:** Elaboración propia

En la parte trasera de la pieza cuenta con orificios de ventilación y dos huecos alrededor de los orificios de los tornillos en los que fijar sus correspondientes tuercas.

Ademas, cuenta con 4 orificios con sus correspondientes alojamientos para tuercas hexagonales con la función de fijar la pieza a la pared.

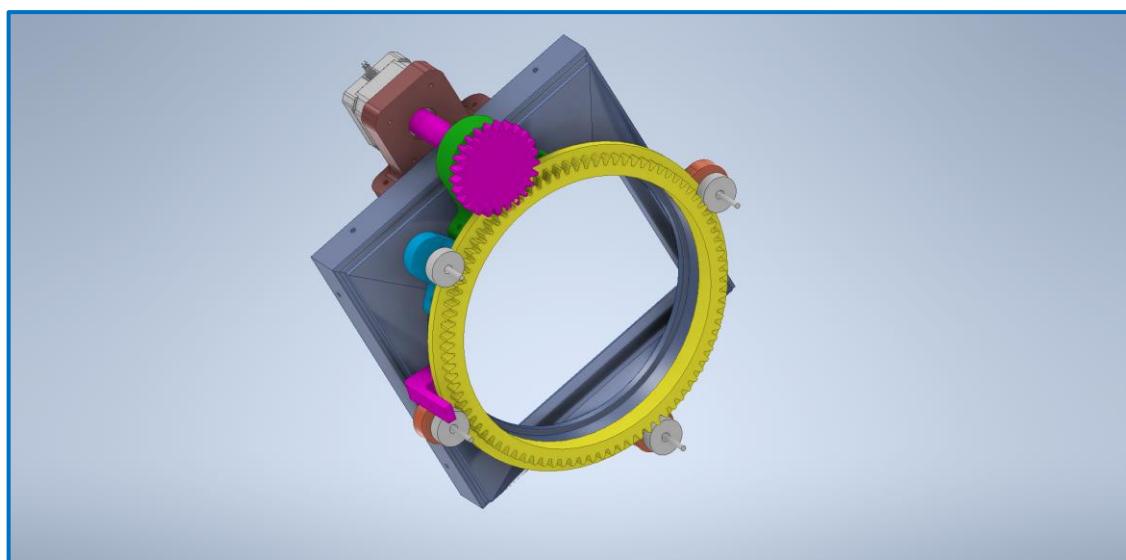
## 6.2. Mecanismo selector

El objetivo de este mecanismo es hacer rotar una estructura que clasifique los residuos en 5 depósitos (metal, papel/cartón, orgánico, plástico y vidrio). Para realizar tal movimiento se empleará un motor “paso a paso”, en este caso el NEMA 17, que hará rotar un engranaje, el cual será orientado con un rodamiento 6202Z para reducir las fuerzas de cortadura. Este engranaje transmitirá el movimiento a uno mayor que estaría fijado a un tubo de PVC. Para soportar el peso de la tubería y facilitar el movimiento de rotación se emplearán otros 4 puntos de soporte, formados por 4 ruedas ancladas en los otros 4 laterales de la pieza intermedia entre el tubo y la parte superior de la máquina.



**Figura 46:** Mecanismo del selector en el contexto de la máquina

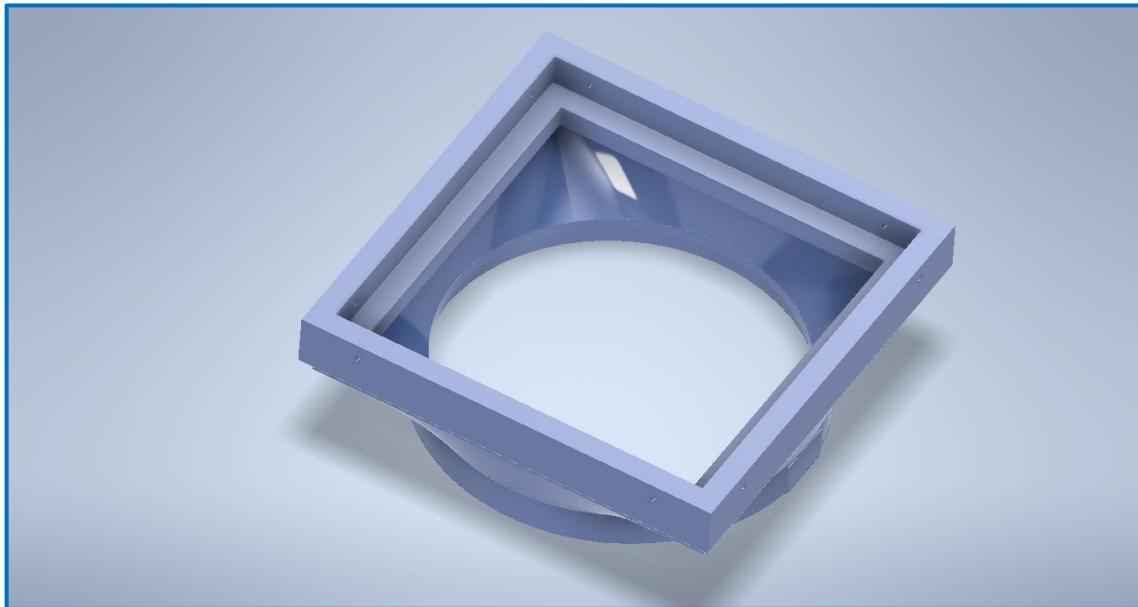
**Fuente:** Elaboración propia



**Figura 47:** Mecanismo del selector. Vista general

**Fuente:** Elaboración propia

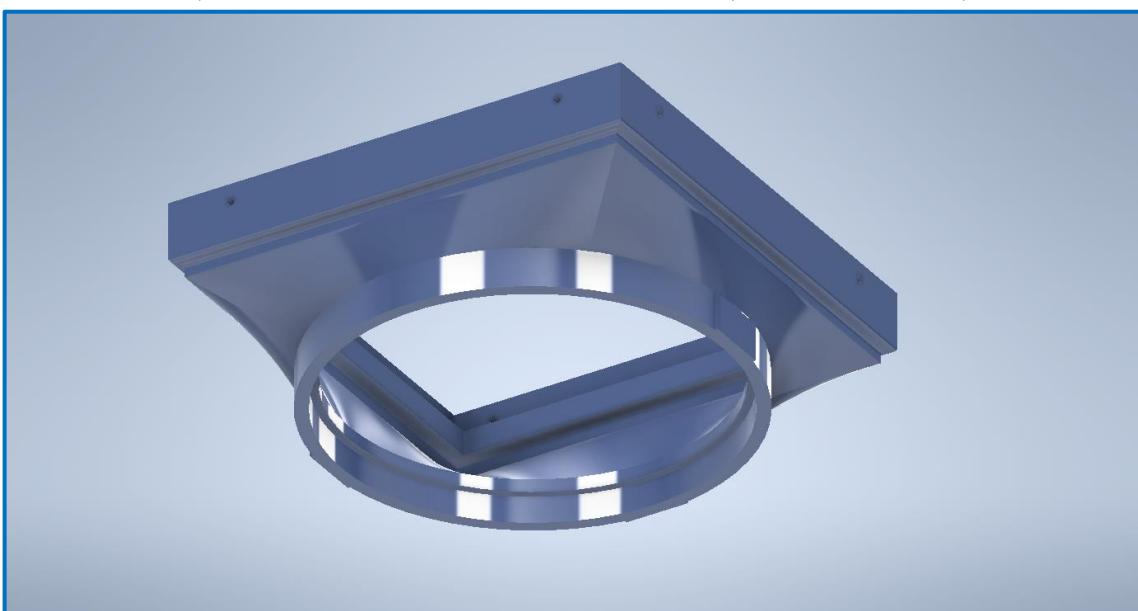
### 6.2.1. Conector tubo con la parte superior de la máquina<sup>17</sup>



**Figura 48:** Conector tubo con la parte superior. Vista superior

**Fuente:** Elaboración propia

Esta pieza permite conectar la parte superior de la máquina de sección cuadrada (160x160x10 mm) con el tubo de PVC de sección circular (diámetro 160 mm).



**Figura 49:** Conector tubo con la parte superior. Vista inferior

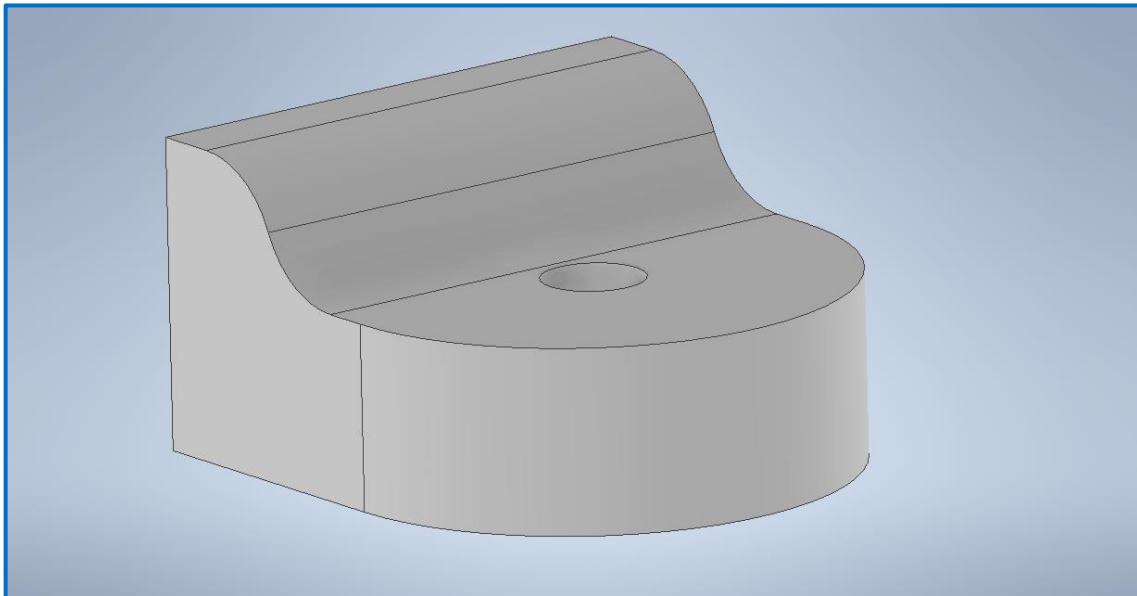
**Fuente:** Elaboración propia

Para facilitar el giro cuenta con una abertura en la que se introduce el tubo.

También cuenta con tres superficies planas situadas de forma paralela a tres de las caras de la máquina, para poder fijar las sujeciones de las ruedas que facilitarán el giro del selector.

<sup>17</sup> Plano en el Anexo 15.1 Planos. Plano 9: Conector tubo parte superior

### 6.2.2. Porta-ruedas<sup>18</sup>



**Figura 50:** Porta-ruedas

**Fuente:** Elaboración propia

Tres de estas piezas se fijarán sobre el conector y se introducirá un tornillo de 3,5 mm en cada una de ellas y una rueda con rodamiento incorporado. El empalme favorece la robustez de la pieza, un factor clave pues el peso del tubo será soportado íntegramente por estas tres piezas.

### 6.2.3. Porta-ruedas auxiliar<sup>19</sup>



**Figura 51:** Porta-ruedas auxiliar

**Fuente:** Elaboración propia

Esta pieza fue diseñada una vez construido el prototipo al valorar que sería recomendable colocar un cuarto apoyo al lado del engranaje conductor.

<sup>18</sup> Plano en el Anexo 15.1 Planos. Plano 10: Sujeción ruedas

<sup>19</sup> Plano en el Anexo 15.1 Planos. Plano 11: Sujeción ruedas auxiliar

#### 6.2.4. Engranaje conducido<sup>20</sup>

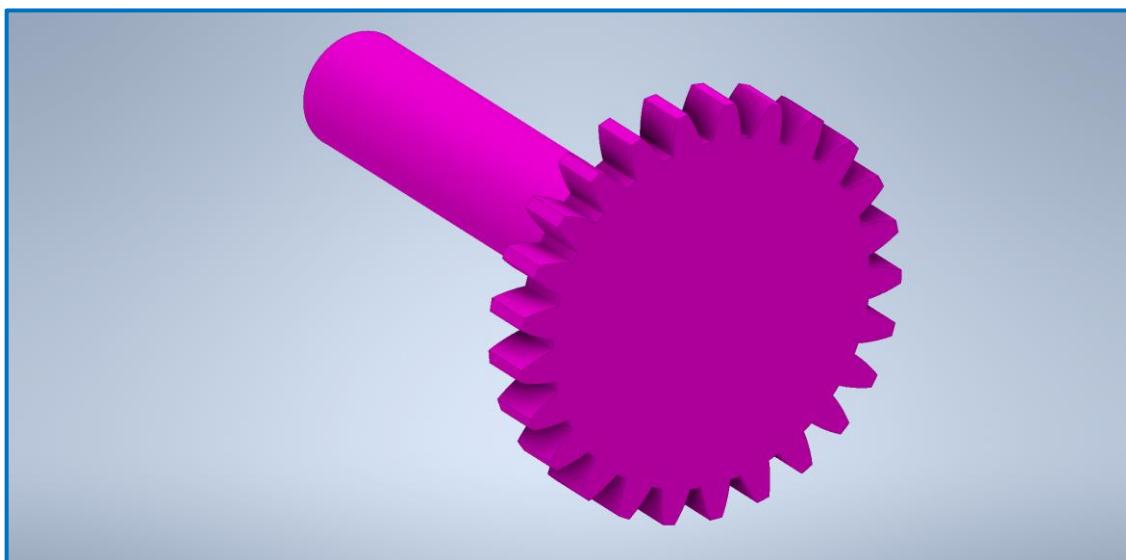


**Figura 52:** Engranaje conducido

**Fuente:** Elaboración propia

Este engranaje de se fija en la superficie de la tubería para transmitirle el movimiento rotativo procedente del motor. Cuenta con un resalte en la parte superior en el que se apoyarán las tres ruedas anteriormente mencionadas.

#### 6.2.5. Engranaje conductor<sup>21</sup>



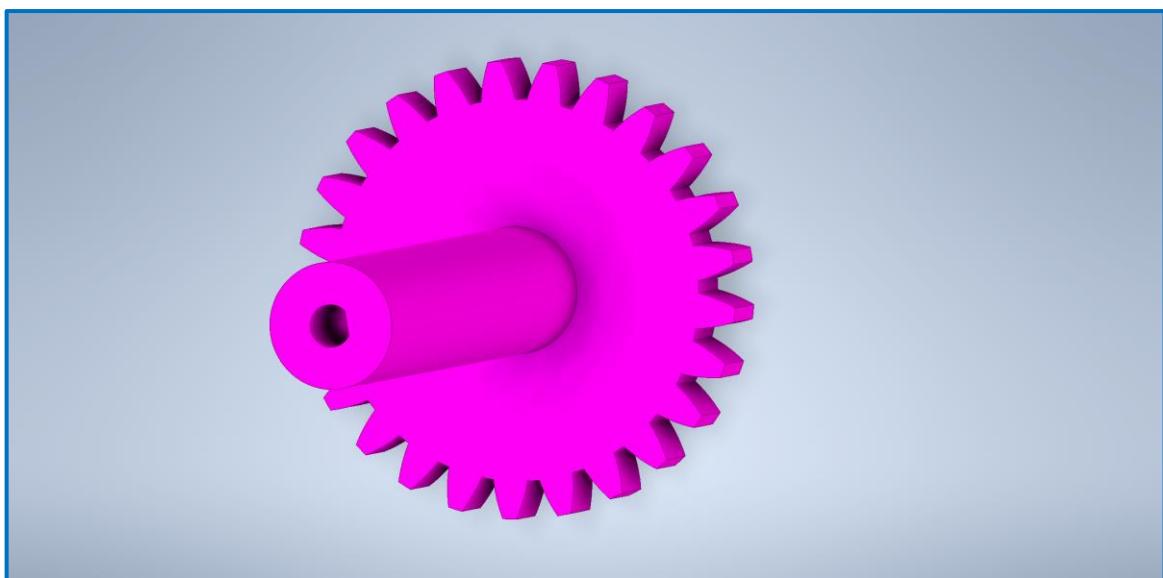
**Figura 53:** Engranaje conductor. Vista general

**Fuente:** Elaboración propia

Este engranaje será el que transmita el movimiento de rotación del NEMA 17 al engranaje conducido y, por tanto, a la tubería. Cuenta con una hendidura en la que introducir el rotor de dicho motor.

<sup>20</sup> Plano en el Anexo 15.1 Planos. Plano 12: Engranaje conducido

<sup>21</sup> Plano en el Anexo 15.1 Planos. Plano 13: Engranaje conductor

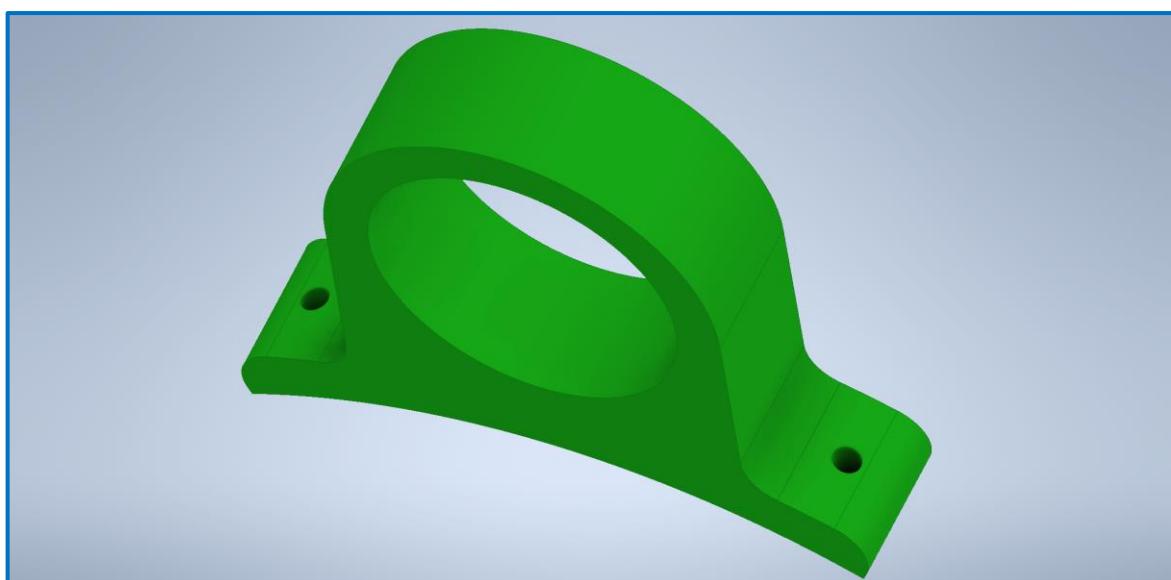


**Figura 54:** Engranaje conductor. Vista de la hendidura para insertar el rotor

**Fuente:** Elaboración propia

#### 6.2.6. Sujeción del rodamiento<sup>22</sup>

El rodamiento 6202Z, que guiará la rotación del engranaje anterior, irá alojado en esta pieza que, además, cuenta con una base circular con el radio de la pieza intermedia para que pueda ser atornillada a esta mediante dos tornillos.

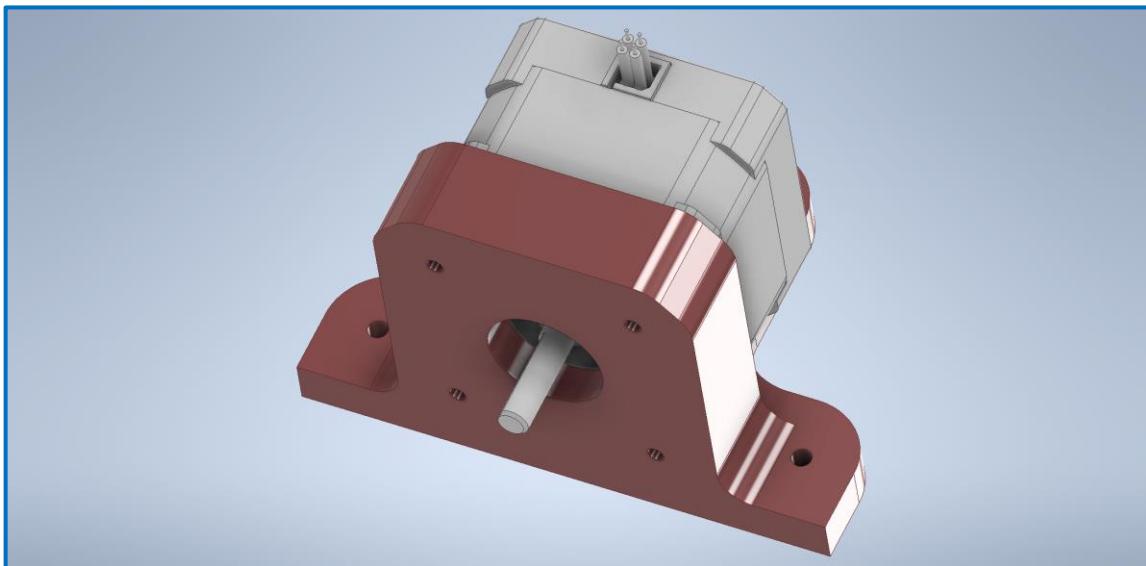


**Figura 55:** Sujeción del rodamiento

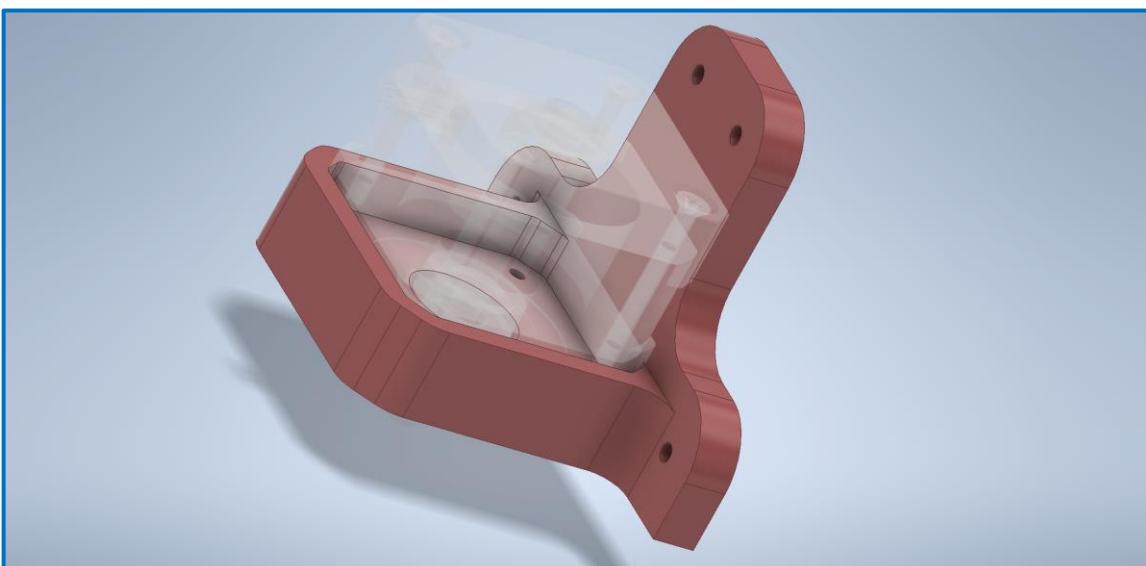
**Fuente:** Elaboración propia

<sup>22</sup> Plano en el Anexo 15.1 Planos. Plano 14: Sujeción del rodamiento

### 6.2.7. Sujeción del motor del selector<sup>23</sup>



**Figura 56:** Sujeción del motor del selector NEMA 17 insertado  
**Fuente:** Elaboración propia

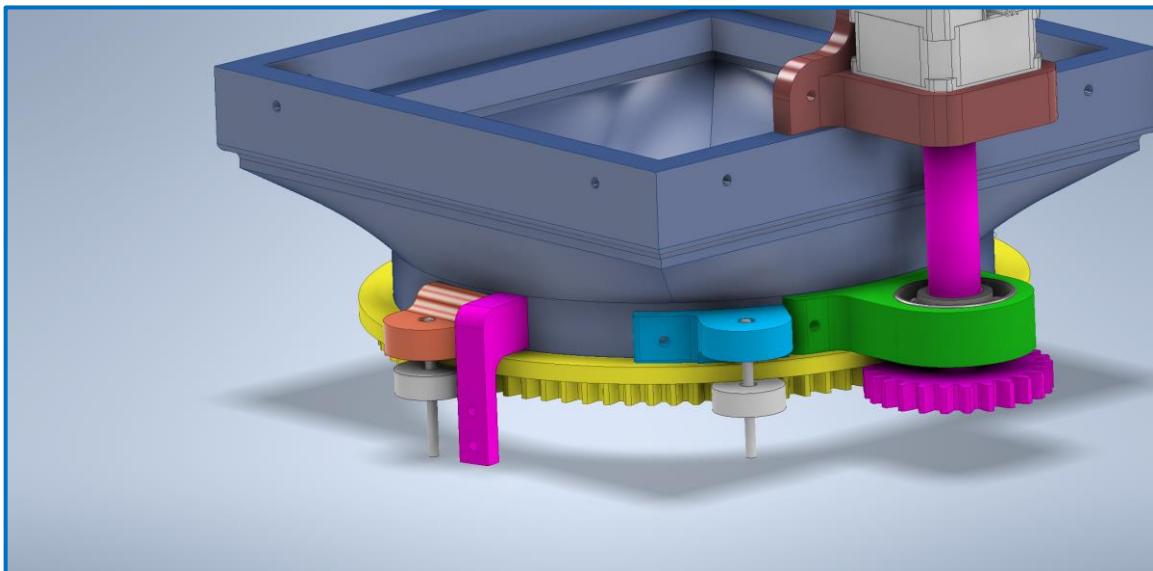


**Figura 57:** Sujeción del motor del selector. Vista de la zona de inserción del motor  
**Fuente:** Elaboración propia

Este componente se encarga de fijar el motor NEMA 17 a la pared de la máquina, mediante 4 tornillos. Se adapta perfectamente a la forma de dicho motor, como se observa en la *Figura 57*.

<sup>23</sup> Plano en el Anexo 15.1 Planos. Plano 15: Sujeción motor selector

### 6.2.8. Sujeción final de carrera del selector<sup>24</sup>



**Figura 58:** Sujeción del final de carrera del selector, en el contexto del mecanismo  
**Fuente:** Elaboración propia

Esta pieza se fijará sobre el conector y servirá de soporte para el final de carrera del selector, para poder realizar un *homing* de este mecanismo.

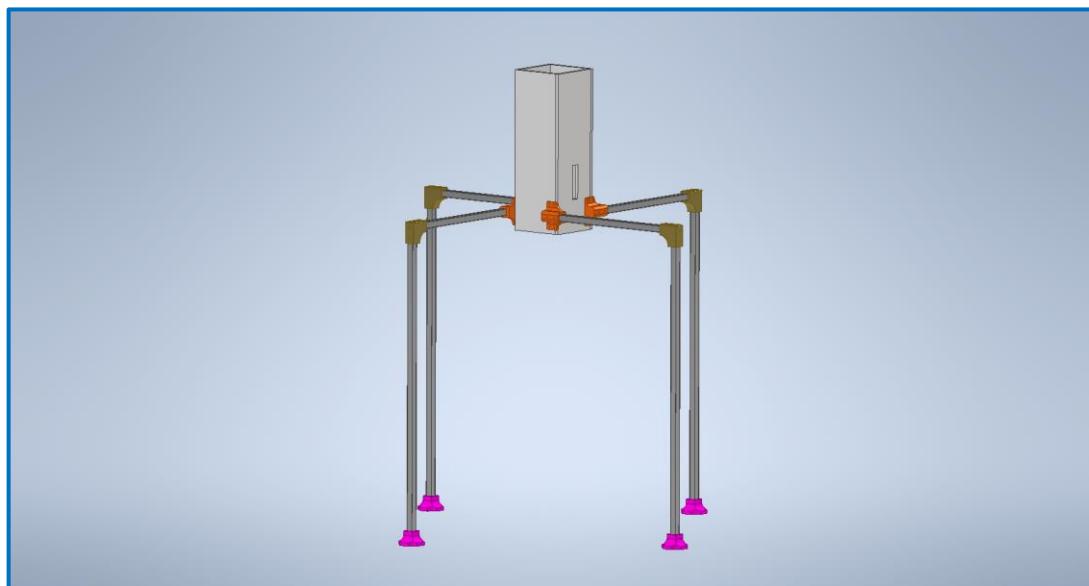


**Figura 59:** Sujeción del final de carrera del selector  
**Fuente:** Elaboración propia

Para fijar el final de carrera se emplearán dos tornillos introducidos en dos orificios de la pieza que se adaptan a la geometría del interruptor.

<sup>24</sup> Plano en el Anexo 15.1 Planos. Plano 16: Sujeción final de carrera

### 6.3. Estructura para el asentamiento de la máquina



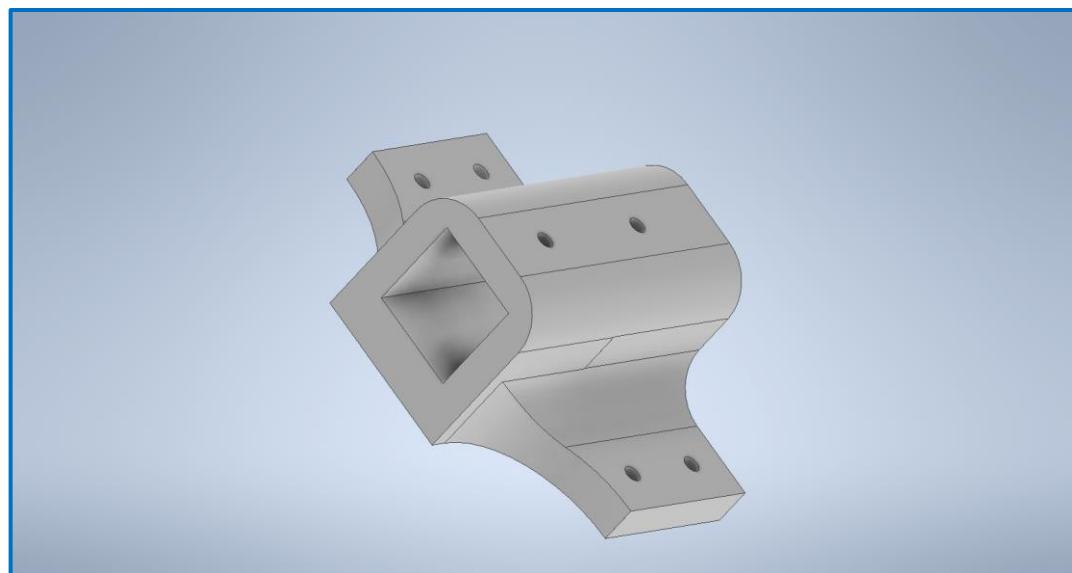
**Figura 60:** Estructura para el asentamiento de la máquina

**Fuente:** Elaboración propia

Para asegurar un asentamiento correcto y seguro de la máquina se decidió colocar cuatro patas formadas por cuatro tubos verticales de 1m para permitir una altura de los depósitos de 440 mm y cuatro tubos horizontales de 0,5 m para permitir el giro del selector, ambos grupos de tubos son de sección cuadrada 20x20 mm.

Para la fijación de dichos tubos a la máquina, entre sí, y al suelo, se diseñaron las tres piezas que describimos en los siguientes epígrafes.

#### 6.3.1. Fijación tubo horizontal<sup>25</sup>



**Figura 61:** Fijación tubo horizontal

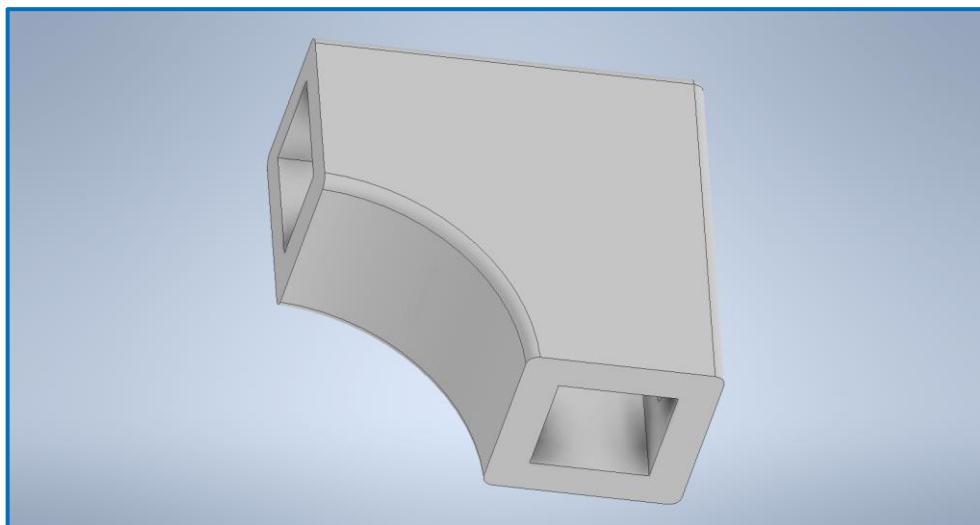
**Fuente:** Elaboración propia

<sup>25</sup> Plano en el Anexo 15.1 Planos. Plano 17: Sujeción tubo horizontal

Esta pieza cumple la función de fijar el tubo horizontal de 0,5 m a la máquina mediante cuatro tornillos de 3 mm, así como fijar el tubo a dicha pieza mediante otros dos tornillos.

El grosor de las paredes, así como los empalmes, garantizan una gran robustez a una pieza critica en la estabilidad de la maquina como esta.

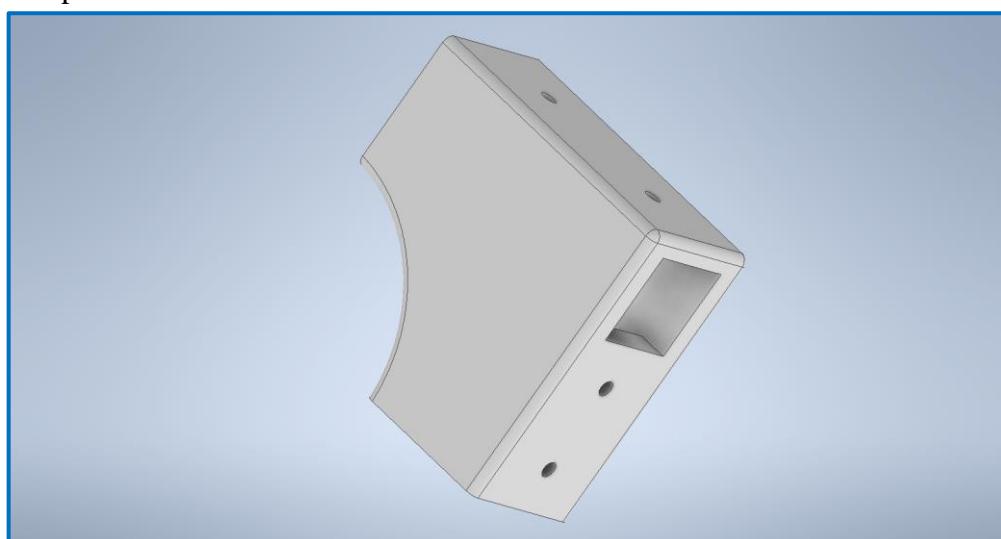
### 6.3.2. Codo 90<sup>o</sup><sup>26</sup>



**Figura 62:** Codo 90°. Vista lateral inferior

**Fuente:** Elaboración propia

Esta pieza une el tubo horizontal con el vertical.



**Figura 63:** Codo 90°. Vista lateral superior

**Fuente:** Elaboración propia

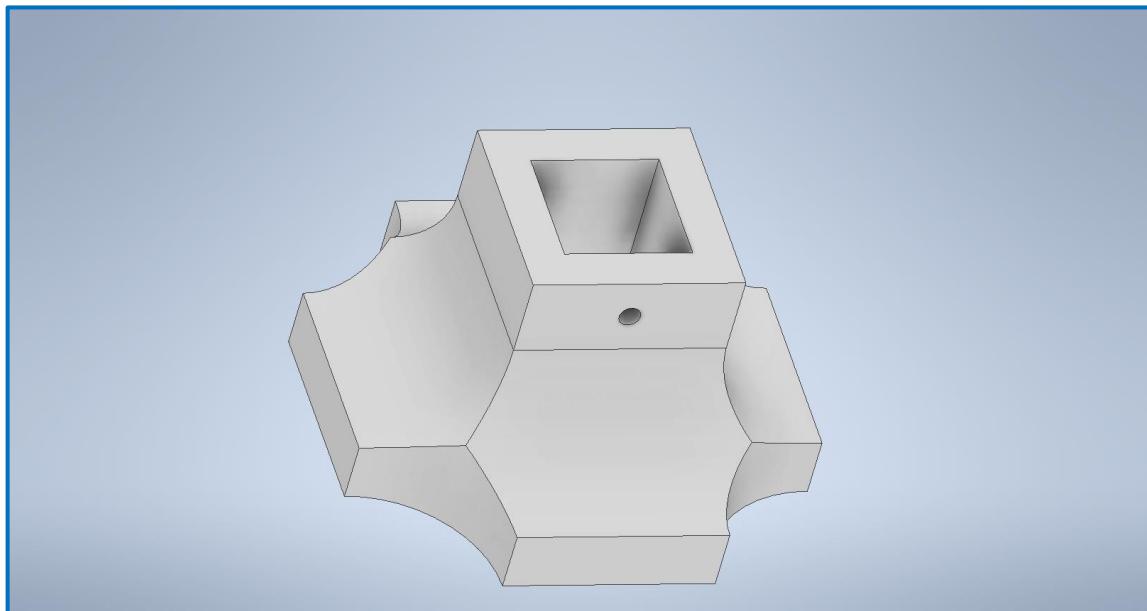
En el caso del tubo vertical este puede atravesar dicha pieza para así ajustar la altura a la que se quiera colocar la máquina.

Cuenta con cuatro orificios, dos para fijar cada uno de los tubos.

Al igual que la pieza anterior, se tuvo en especial consideración la solidez de la pieza.

<sup>26</sup> Plano en el Anexo 15.1 Planos. Plano 18: Codo 90°

### 6.3.3. Pie<sup>27</sup>



**Figura 64:** Pie

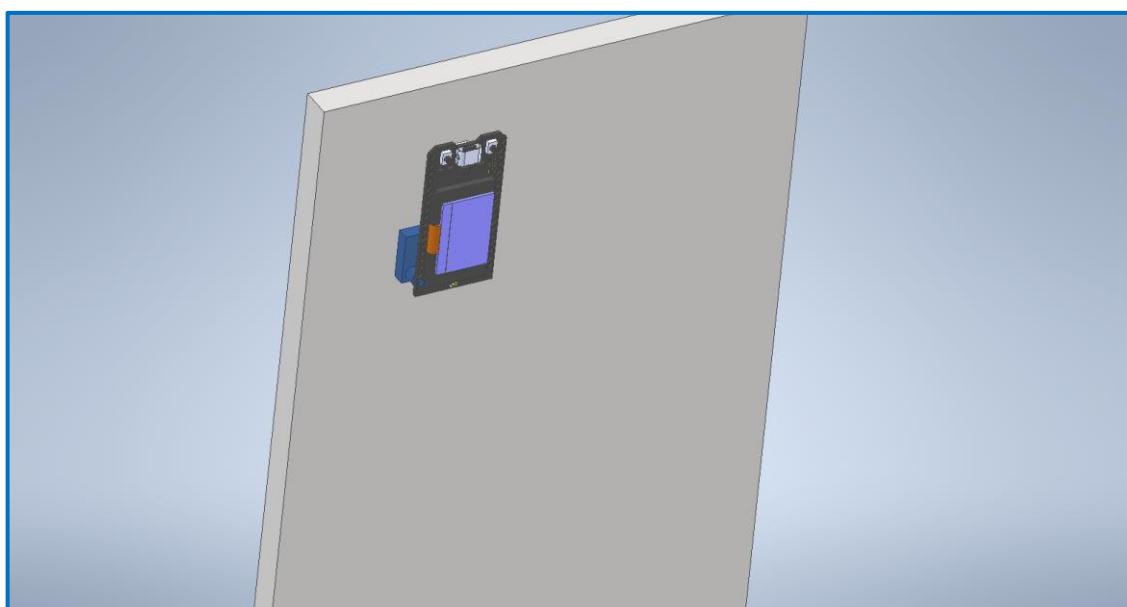
**Fuente:** Elaboración propia

Esta pieza garantiza a la máquina una superficie estable sobre la que apoyarse.

Cuenta con un orificio para fijar el tubo correspondiente.

## 6.4. Otras piezas

### 6.4.1. Sujeción ESP32 (LoRa)<sup>28</sup>



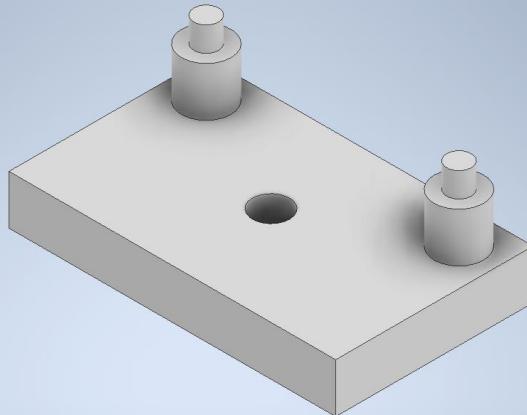
**Figura 65:** Sujeción ESP32. ESP32 insertado

**Fuente:** Elaboración propia

<sup>27</sup> Plano en el Anexo 15.1 Planos. Plano 19: Pie

<sup>28</sup> Plano en el Anexo 15.1 Planos. Plano 20: Sujeción ESP32

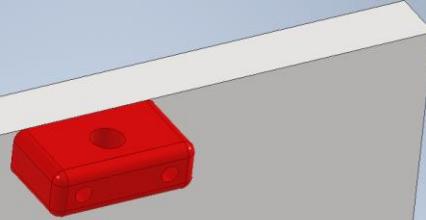
El ESP32 encargado de las transmisiones LoRa se coloca sobre esta pieza.



**Figura 66:** Sujeción ESP32  
**Fuente:** Elaboración propia

La forma de esta pieza se adapta a los dos orificios que tiene el ESP32 para así poder ofrecer una correcta sujeción.

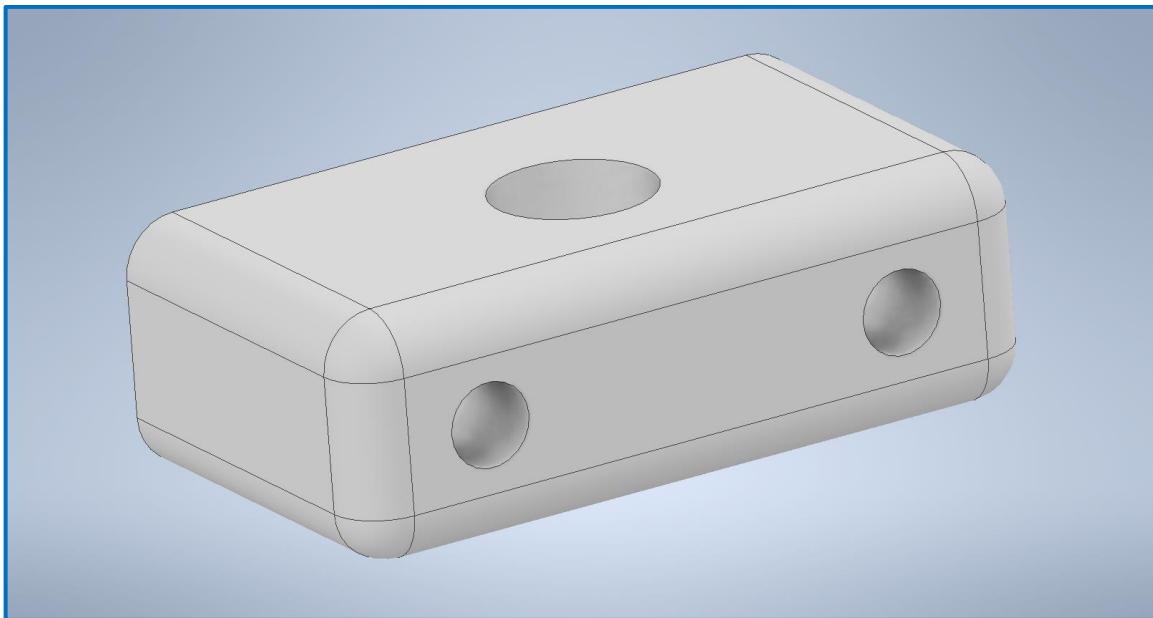
#### 6.4.2. Sujeción antena LoRa<sup>29</sup>



**Figura 67:** Sujeción antena LoRa. Vista en el contexto de la máquina  
**Fuente:** Elaboración propia

La antena del ESP32 se sujetará mediante esta pieza en la parte superior de la máquina.

<sup>29</sup> Plano en el Anexo 15.1 Planos. Plano 21: Sujeción antena

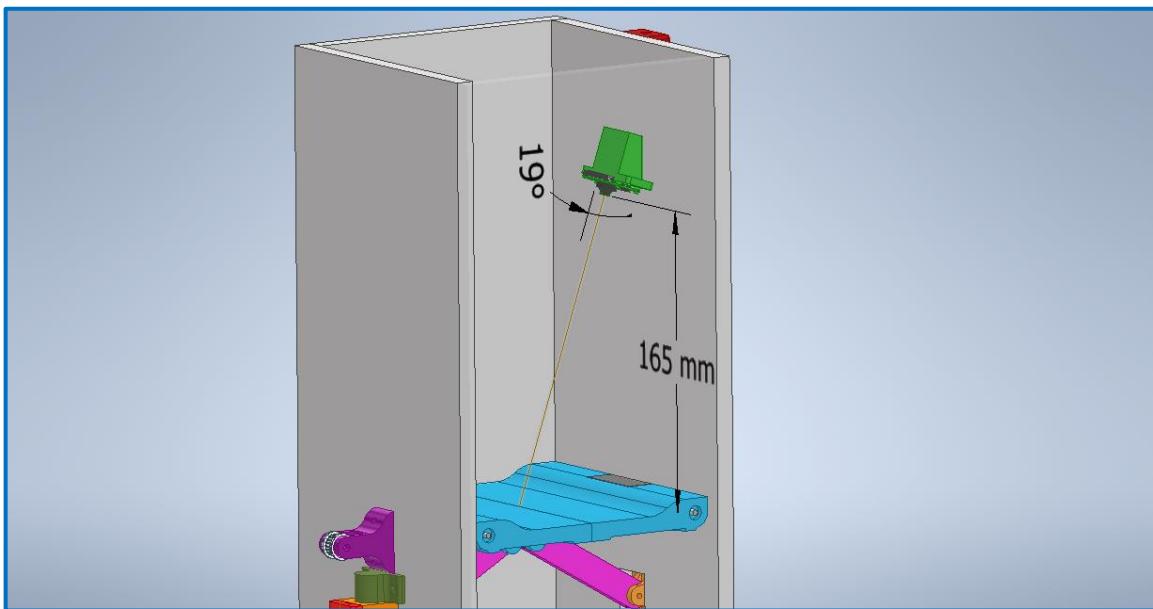


**Figura 68:** Sujeción antena LoRa

**Fuente:** Elaboración propia

Cuenta con un orificio vertical de 6 mm en el que se introduce la antena y dos orificios horizontales en los que atornillar la pieza a la máquina.

#### 6.4.3. Sujeción cámara<sup>30</sup>



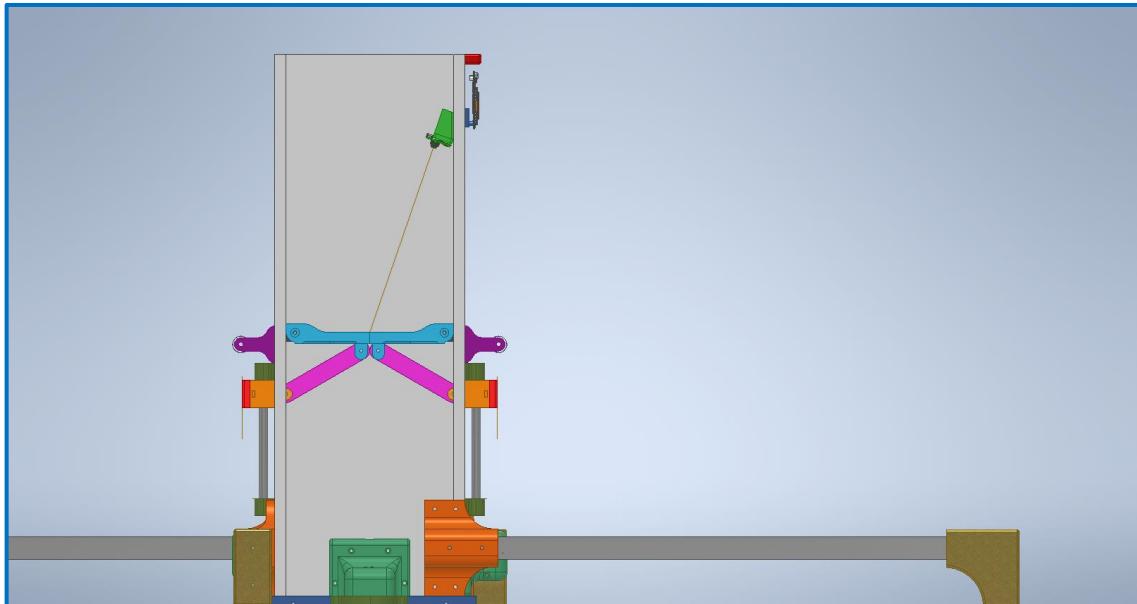
**Figura 69:** Sujeción cámara. Colocación y ángulo

**Fuente:** Elaboración propia

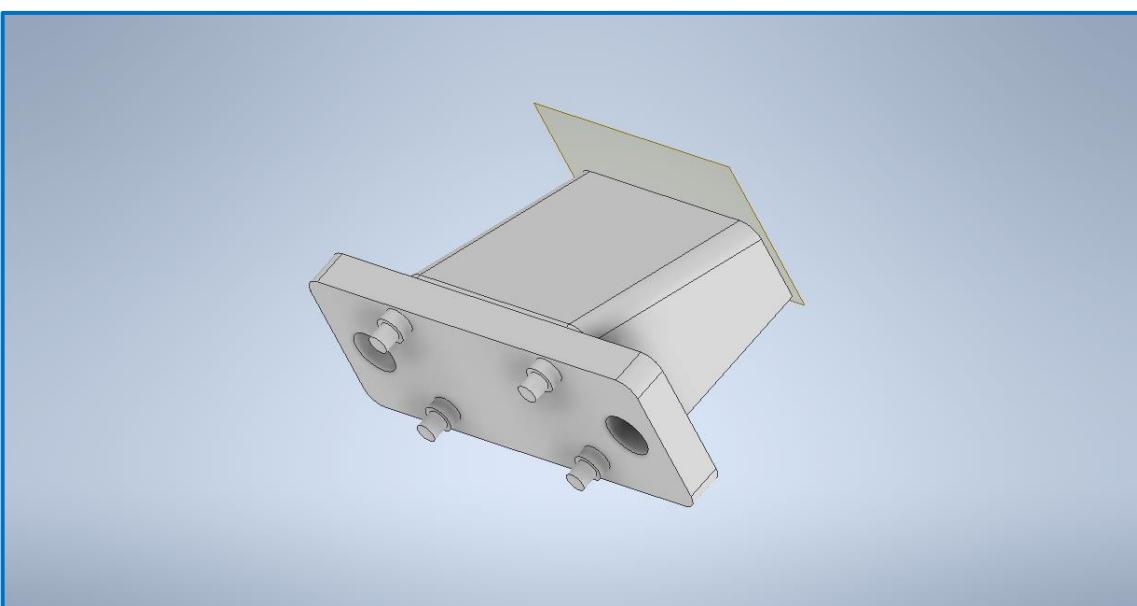
La colocación de la cámara es clave para obtener unas buenas imágenes en las que basar la posterior clasificación. Por ello, para diseñar la pieza que la sujetaba, era necesario tener en cuenta el tamaño máximo de los residuos a tratar, especialmente la altura (ya que la anchura ya viene limitada por la apertura de las compuertas de 76,46 mm). Como se expresó anteriormente,

<sup>30</sup> Plano en el Anexo 15.1 Planos. Plano 22: Sujeción cámara

el tamaño máximo del residuo para el que fue diseñada esta máquina podría ser una botella pequeña de plástico de unos 150 mm de altura. Por tanto, dejando un margen de 15 mm entre el sensor de la cámara y la parte superior del residuo se decidió colocar la cámara a una altura de 165 mm respecto a la base de la compuerta. Teniendo este dato, el eje del sensor debía de formar  $19^\circ$  con la pared de la máquina para poder centrarla en el punto medio de las dos compuertas.



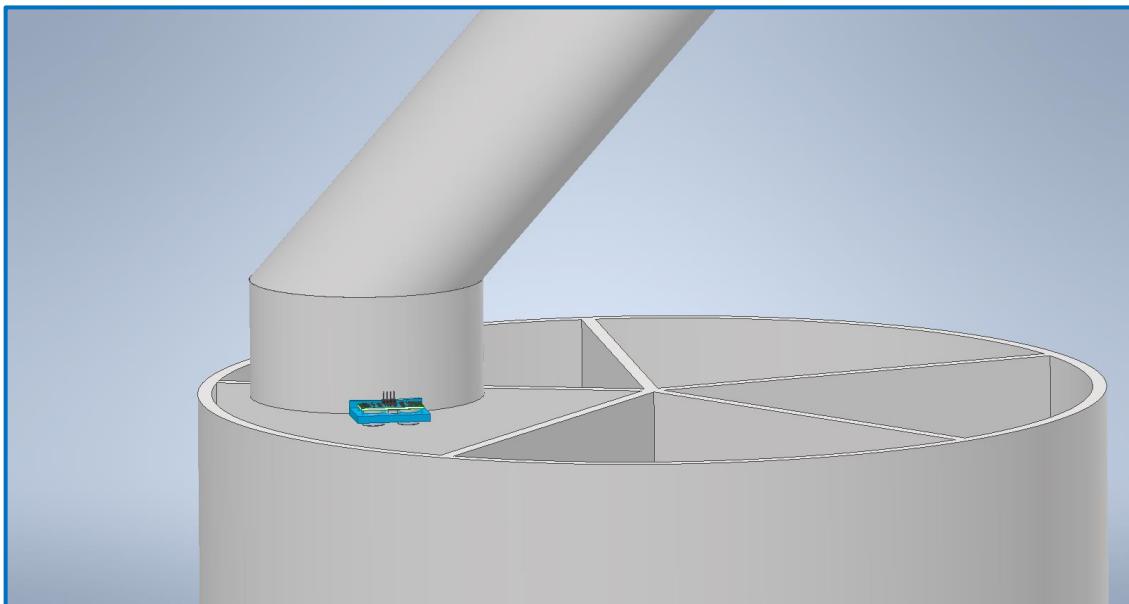
**Figura 70:** Sujeción de la cámara. Colocación vista de perfil  
**Fuente:** Elaboración propia



**Figura 71:** Sujeción de la cámara  
**Fuente:** Elaboración propia

Sabiendo el ángulo, así como las dimensiones de la cámara y de sus cuatro orificios de sujeción, se diseñó esta pieza que, además, cuenta con dos orificios en los que situar dos LEDs de alta potencia para iluminar la escena.

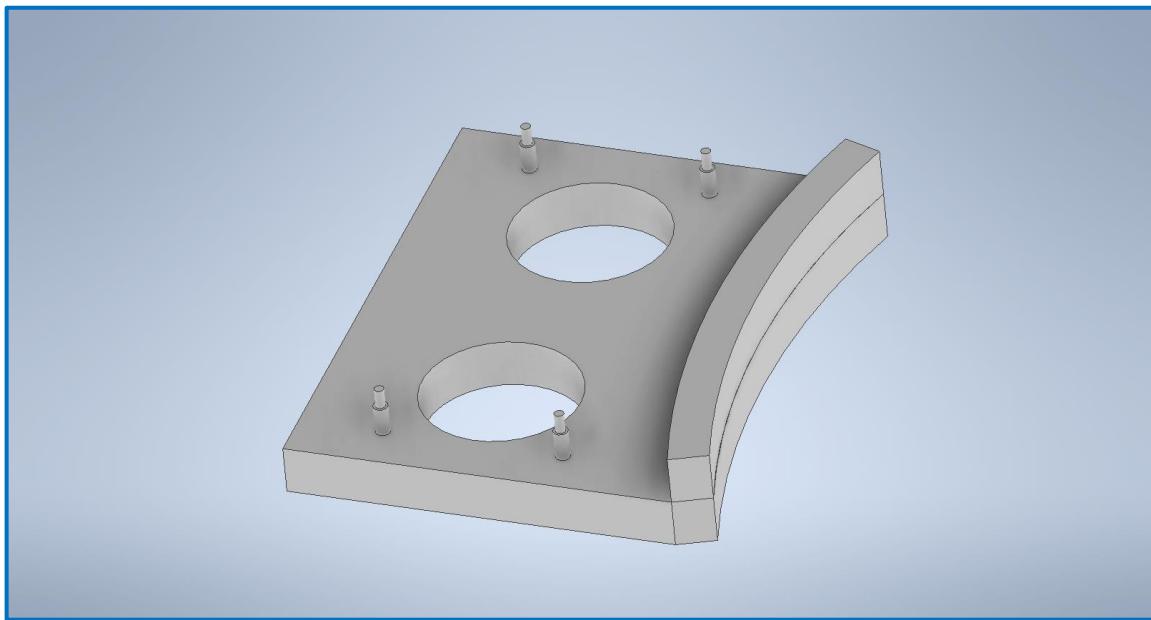
#### 6.4.4. Sujeción sensor de ultrasonidos<sup>31</sup>



**Figura 72:** Sujeción del sensor de ultrasonidos, en el contexto de la máquina

**Fuente:** Elaboración propia

Para medir el nivel de cada depósito se empleará un sensor de ultrasonido para fijarlo a la parte inferior del tubo y para ello se diseñó esta pieza.



**Figura 73:** Sujeción del sensor de ultrasonidos

**Fuente:** Elaboración propia

La pieza está diseñada para adaptarse a los cuatro orificios de sujeción del sensor hc-sr04 y a los dos cilindros que forman el emisor y receptor de dicho sensor.

---

<sup>31</sup> Plano en el Anexo 15.1 Planos. Plano 23: Sujeción ultrasonidos

#### 6.4.5. Plantilla para serigrafía



**Figura 74:** Plantilla para serigraffía

**Fuente:** Elaboración propia

Para serigrafiar con espray el nombre de la máquina se diseñó una lámina con el nombre “Smartbin”.

## 7. IMPRESIÓN 3D Y CONSTRUCCIÓN DEL PROTOTIPO

### 7.1. Impresión 3D

El proceso de impresión de las 46 piezas que componen la máquina se realizó en una impresora de modelado por deposición fundida de la marca ANET y modelo E12, pero profundamente modificada y propiedad del autor de este trabajo. Cuenta con una superficie de impresión de 300x300x400 mm.

Para la impresión de todas las piezas de este trabajo se ha empleado PLA reciclado de 1,75 mm de diámetro.

#### 7.1.1. Parámetros de impresión

A continuación, se adjunta una tabla con los parámetros más importantes de impresión de cada pieza:

**Tabla 2:** Parámetros y tiempos de impresión

**Fuente:** Elaboración propia

| Pieza                        | Diam. nozzle (mm) | Altura de capa (mm) | Nº lineas de pared | Relleno (%) | Tipo de relleno | Soporte | Tiempo de impresión por ud | Uds    | Tiempo de impresión |
|------------------------------|-------------------|---------------------|--------------------|-------------|-----------------|---------|----------------------------|--------|---------------------|
| Sujeción compuerta           | 0.4               | 0.2                 | 6                  | 100         | Lineas          | SI      | 1:14:00                    | 2      | 2:28:00             |
| Compuerta                    | 0.4               | 0.3                 | 6                  | 30          | Tri-hexagonal   | SI      | 7:34:00                    | 2      | 15:08:00            |
| Eslabón                      | 0.4               | 0.2                 | 6                  | 60          | Tri-hexagonal   | SI      | 0:40:00                    | 2      | 1:20:00             |
| Embolo                       | 0.4               | 0.2                 | 6                  | 60          | Tri-hexagonal   | SI      | 1:39:00                    | 2      | 3:18:00             |
| Sujeción varilla             | 0.4               | 0.2                 | 6                  | 60          | Tri-hexagonal   | SI      | 0:41:00                    | 4      | 2:44:00             |
| Sujeción correa              | 0.2               | 0.1                 | 6                  | 100         | Lineas          | SI      | 2:15:00                    | 2      | 4:30:00             |
| Porta-poleas                 | 0.4               | 0.2                 | 6                  | 60          | Tri-hexagonal   | SI      | 0:57:00                    | 2      | 1:54:00             |
| Sujeción motor compuertas    | 0.4               | 0.2                 | 4                  | 60          | Rejilla         | SI      | 4:53:00                    | 3      | 14:39:00            |
| Conector tubo                | 1                 | 0.4                 | 4                  | 25          | Lineas          | SI      | 10:36:00                   | 1      | 10:36:00            |
| Porta ruedas                 | 0.4               | 0.2                 | 4                  | 60          | Tri-hexagonal   | SI      | 0:40:00                    | 3      | 2:00:00             |
| Porta ruedas auxiliar        | 0.4               | 0.2                 | 4                  | 60          | Tri-hexagonal   | SI      | 0:45:00                    | 1      | 0:45:00             |
| Engranaje conducido          | 0.4               | 0.2                 | 4                  | 60          | Tri-hexagonal   | NO      | 7:36:00                    | 1      | 7:36:00             |
| Engranaje conductor          | 0.4               | 0.2                 | 4                  | 60          | Tri-hexagonal   | NO      | 2:32:00                    | 1      | 2:32:00             |
| Sujeción del rodamiento      | 0.5               | 0.3                 | 4                  | 60          | Tri-hexagonal   | SI      | 1:07:00                    | 1      | 1:07:00             |
| Suj. del motor del selector  | 0.5               | 0.3                 | 6                  | 60          | Tri-hexagonal   | NO      | 2:20:00                    | 1      | 2:20:00             |
| Sujeción final de carrera    | 0.5               | 0.2                 | 6                  | 60          | Tri-hexagonal   | NO      | 0:19:00                    | 1      | 0:19:00             |
| Fijacion tubo horizontal     | 0.5               | 0.3                 | 6                  | 60          | Tri-hexagonal   | SI      | 3:40:00                    | 4      | 14:40:00            |
| Codo 90º                     | 0.5               | 0.3                 | 6                  | 60          | Tri-hexagonal   | SI      | 3:30:00                    | 4      | 14:00:00            |
| Pie                          | 0.5               | 0.3                 | 6                  | 60          | Tri-hexagonal   | SI      | 2:48:00                    | 4      | 11:12:00            |
| Sujecion ESP32               | 0.4               | 0.2                 | 4                  | 40          | Lineas          | NO      | 0:13:00                    | 1      | 0:13:00             |
| Sujecion antena LoRa         | 0.4               | 0.2                 | 4                  | 40          | Lineas          | SI      | 0:27:00                    | 1      | 0:27:00             |
| Sujeción camara              | 0.4               | 0.2                 | 4                  | 40          | Lineas          | SI      | 0:51:00                    | 1      | 0:51:00             |
| Sujeción sensor ultrasonidos | 0.4               | 0.2                 | 4                  | 40          | Lineas          | NO      | 0:50:00                    | 1      | 0:50:00             |
| Serigrafía                   | 0.5               | 0.3                 | 4                  | 100         | Lineas          | NO      | 0:20:00                    | 1      | 0:20:00             |
|                              |                   |                     |                    |             |                 |         |                            | Piezas | 46                  |

- Diámetro del Nozzle

El diámetro del nozzle es la apertura del orificio por el que se deposita el plástico fundido. De este parámetro depende la rapidez de la impresión, su respuesta a esfuerzos mecánicos. Es por esto que se han empleado diferentes diámetros para cada pieza. Mayoritariamente se ha empleado el diámetro 0,4 mm, puesto que es un diámetro muy común, que vale para piezas de un tamaño estándar y que permite una impresión rápida, se ha tomado este diámetro como valor de referencia para diseñar las piezas, como se indicó en el epígrafe 6, donde se aborda el diseño del prototipo. Es necesario tener en cuenta este parámetro para diseñar una pieza con dimensiones que al dividirlas por el diámetro del nozzle den como resultado un número entero de capas. Pero en el caso de la pieza que pinza la correa GT2 para fijarla al émbolo se ha empleado un diámetro de 0,2 mm para garantizar una impresión de buena calidad. En las piezas encargadas del asentamiento, así como en la pieza que aloja el rodamiento del engranaje del selector se ha empleado un diámetro de 0,5 mm, pues son piezas con un elevado estrés mecánico. Por último, en la pieza que conecta el tubo con el resto de la máquina se ha impreso con un diámetro de 1 mm, pues es una pieza muy grande y era necesario reducir el tiempo de impresión.

- Altura de capa

Este parámetro determina la altura entre capa y capa de la impresión 3D, por tanto, es determinante en cuanto a velocidad de impresión y calidad superficial de la pieza. Mayoritariamente se ha empleado una altura de capa de 0,2 mm, a excepción de la sujeción para la correa, que se empleó una altura de 0,1 mm por una mejor calidad de una pieza con detalles de pequeño tamaño como es el perfil de la correa GT2. Con relación a la compuerta y los elementos de asentamiento, se imprimieron con un diámetro de 0,3 mm pues la calidad superficial es irrelevante en estas piezas y una altura de capa de 0,4 mm, en el caso del conector, para aumentar la velocidad de impresión.

- Número de líneas de pared

En general, las piezas de pequeño tamaño y con orificios cercanos a los bordes se han impreso con seis líneas de pared. De esta forma se aseguraba que en ese espacio entre el orificio y el borde de la pieza la impresión fuese al 100% de relleno o, en las piezas en las que es preferible un contorno resistente y un interior, con poco relleno. Por ejemplo, el caso de la compuerta cuenta con un resalte en el que se conecta el eslabón y es aconsejable que no haya relleno en esa parte de la pieza para que sea una zona resistente pues va a estar sometida a diferentes fuerzas, pero tampoco sería recomendable que toda la pieza se imprimiese al 100% de relleno por eso se imprimió con seis líneas de pared. El resto de las piezas que no tenían estos requerimientos se imprimieron con cuatro líneas.

- Densidad de relleno y tipo

Las piezas que pueden ser sometidas a estrés mecánico se imprimieron con un relleno de tipo tri-hexagonal, las que no, con relleno en líneas. En el caso de la sujeción de la puerta se

imprimió con un 100% de relleno porque es una pieza pequeña y que cuenta con numerosos orificios.

- Soporte

El diseño de las piezas y la colocación de estas en el plato de impresión se ha hecho de forma que redujese al máximo el uso de soportes, sobre todo en piezas en las que la calidad superficial es importante, como la sujeción de la correa o el émbolo.

No obstante, cuando existen voladizos considerables en una pieza es necesario recurrir a ello.

- Tiempos de impresión

Los tiempos de impresión se han obtenido imprimiendo las piezas en la posición óptima, de forma que se redujesen al máximo los soportes y el tiempo de impresión, y a una velocidad media de los movimientos de la impresora de 70 mm/s (velocidad de relleno de 140 mm/s, velocidad de pared de 35 mm/s y velocidad de posicionamiento de 175 mm/s). Si bien en el momento de la impresión se ha modificado en algunas piezas la velocidad, ya fuese aumentándola o disminuyéndola.

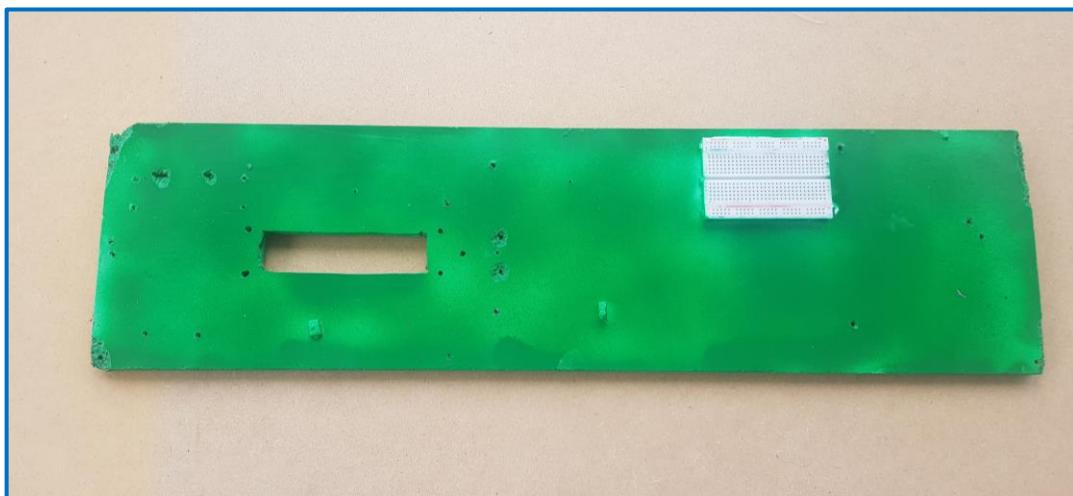
## 7.2. Construcción



**Figura 75:** Vista general de la maquina ensamblada  
**Fuente:** Elaboración propia

El proceso de construcción fue gradual, a medida que se imprimían las piezas y, en algunos casos, la progresiva construcción del proyecto influyó también en el diseño de nuevas piezas o mejoras.

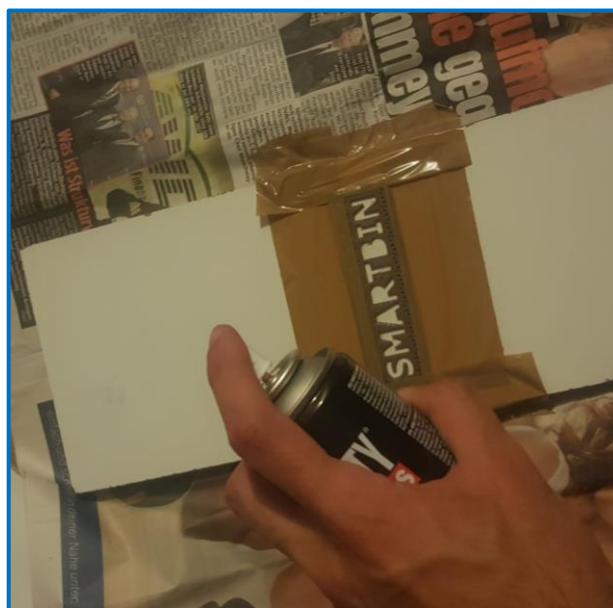
Se comenzó cortando con una sierra de calar un tablero de aglomerado de 60x120x10cm en cuatro segmentos de 60x16 cm para conformar las paredes de la máquina. A dos de ellas se le efectuaron los orificios necesarios para el desplazamiento del émbolo, según las medidas acotadas en el Plano 26 (Pared Compuerta) que se adjunta en el Anexo de Planos. Posteriormente se pintaron con un espray verde la cara visible, es decir la cara exterior de cada pared, dejando los tonos blancos que producía la propia melanina original del tablero.



**Figura 76:** Pared de la maquina pintada

**Fuente:** Elaboración propia

También se serigrafió el logo descrito en el epígrafe 6.4.5, con un espray de color rojo.

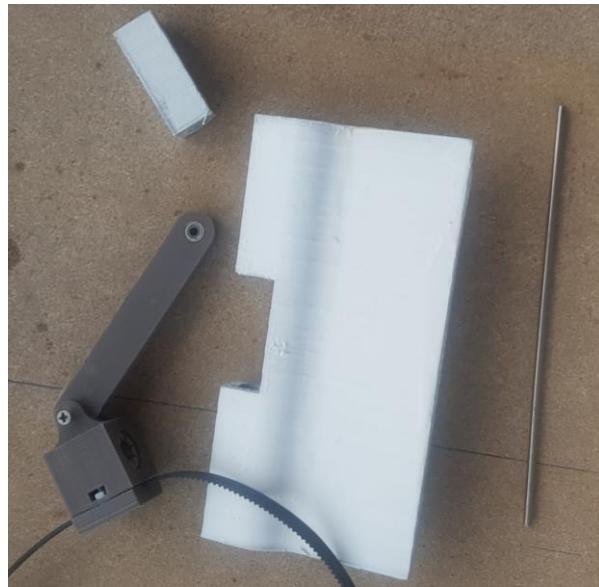


**Figura 77:** Proceso de pintado de la serigrafía

**Fuente:** Elaboración propia

Se pintaron de blanco las compuertas y las sujetaciones de estas, pues se habían impreso inicialmente en un tono gris, pero se consideró que sería más recomendable que, al menos la

superficie visible para la cámara, fuese blanca para que el fondo de las imágenes tuviese este color, como el de las imágenes del set de entrenamiento.



**Figura 78:** Compuerta y sujeción pintadas  
**Fuente:** Elaboración propia

A continuación, se procedió al ensamblaje del mecanismo de las compuertas. Primero se introdujeron los cuatro rodamientos por las compuertas, se unió cada compuerta a su sujeción mediante las varillas de 3 mm y se fijaron las sujetaciones a cada pared de la máquina atendiendo a las medidas respecto a los orificios, calculadas en el epígrafe 6.1. Luego se introdujeron dos rodamientos por cada eslabón y se fijaron a la compuerta y al émbolo mediante tornillos.



**Figura 79:** Sujeción, compuerta, eslabón, émbolo y sujeción de la correa ensamblados  
**Fuente:** Elaboración propia

Posteriormente se cortaron dos barras circulares de aluminio de 50,3 mm de longitud. Estas barras se introdujeron en cada uno de los IGUs de cada émbolo y se fijaron a las paredes de la máquina mediante las dos sujeciones atornilladas en cada extremo del orificio. En las sujeciones superiores de cada varilla se atornilló también un final de carrera que en el siguiente epígrafe se comentará con mayor profundidad.

A continuación, se introdujeron los motores “paso a paso” en sus respectivas sujeciones y se atornillaron a las paredes de la máquina. En el lado opuesto se fijaron, mediante tornillos, las piezas porta-poleas. Teniendo esta configuración se midió la longitud necesaria para correa dentada y se seccionó en consecuencia. Y, por último, se fijó la correa al émbolo mediante la pieza descrita en el epígrafe 6.1.5 (sujeción correa).



**Figura 80:** Vista exterior del mecanismo de la compuerta ensamblado  
**Fuente:** Elaboración propia

Una vez ensamblados los dos mecanismos de las compuertas se atornillaron las cuatro paredes a la pieza intermedia para conectar con el tubo selector.

Después se montaron las estructuras de soporte de la máquina. Se cortaron con una sierra de calar dos tubos de 1 m a la mitad, es decir formando cuatro tubos de 50 cm para que hiciesen de soporte horizontal. Por tanto, se conectaron mediante los codos de 90° diseñados a otros cuatro tubos de 1 metro de largo a modo de soporte vertical. Las barras horizontales se introdujeron en el codo hasta el fondo de forma que las barras verticales se apoyasen sobre ellas, aunque si posteriormente se decide bajar el prototipo se podrían volver a ajustar pues el

diseño del codo así lo permite. También se fijaron los pies a cada barra vertical mediante un tornillo.

En la pieza intermedia se fijaron con pegamento termofusible las sujetaciones de las ruedas o porta-ruedas.

Se unieron, con pegamento de PVC, dos codos de 45° y 160 mm de diámetro para formar el selector. A su vez, se adhirió dicho tubo al engranaje conducido mediante pegamento termofusible.

Una vez hecho esto, se colocó la tubería a la altura de la pieza intermedia y de las sujetaciones para las ruedas, y se colocaron las cuatro ruedas para que soportasen el peso del selector.

A continuación, se procedió al ensamblaje del mecanismo del selector. Para ello se introdujo el rodamiento para guiar el engranaje conductor dentro de la pieza diseñada para alojarlo. Se colocó el engranaje conductor encajándolo junto al conducido y se fijó el soporte del rodamiento a la pieza intermedia mediante dos tornillos. Posteriormente se colocó el NEMA 17 en su sujeción, se fijó con 4 tornillos esta sujeción a la superficie de la pared y se introdujo el rotor de dicho motor en la abertura que presenta el engranaje conductor para tal efecto. Así, fue posible fijar la sujeción del NEMA a la pared de la máquina en la posición correcta.



**Figura 81:** Vista del mecanismo del selector ensamblado  
**Fuente:** Elaboración propia

Se fijó la sujeción de la cámara en la posición calculada en el epígrafe 6.4.3 mediante pegamento termofusible.



**Figura 82:** Montaje de la cámara

**Fuente:** Elaboración propia

También se atornilló la sujeción del LoRa mediante un tornillo y la sujeción de la antena mediante otros dos tornillos. La sujeción del ultrasonido también se colocó en la parte baja del tubo con termofusible.

Por último, se conectó toda la electrónica siguiendo los esquemas que se presentarán en el siguiente epígrafe.



**Figura 83:** Visión general de la electrónica montada

**Fuente:** Elaboración propia

## 8. COMPONENTES ELECTRÓNICOS. ESTRUCTURA DE CONEXIONES

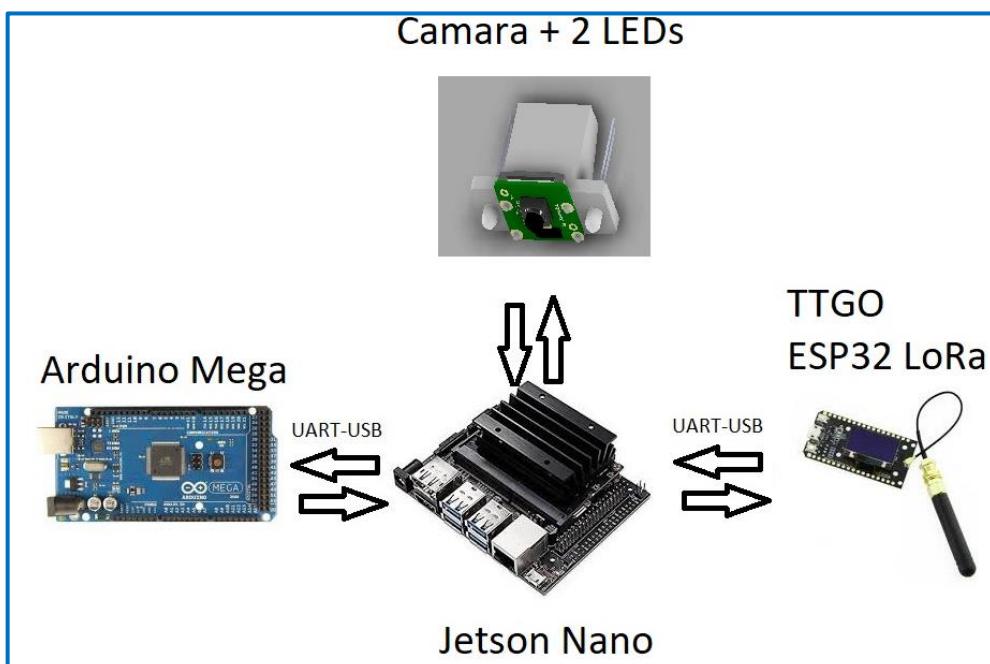
Como se mencionó en el epígrafe 5.4 relativo a la congelación del modelo e inferencia, para la ejecución de la red neuronal, así como para la gestión de los distintos subprocessos se optó por emplear una placa de Nvidia® denominada Jetson Nano® pues cuenta con una GPU y una capacidad de procesamiento mayor que un Arduino o una Raspberry.

Para el control de los movimientos se empleó un Arduino Mega por ser este un proceso que no requiere grandes procesamientos, pero si una cantidad elevada de terminales de E/S. Con esto, además, se libera a la Jetson Nano de carga de trabajo.

Para la comunicación LoRa se emplean dos ESP32, uno colocado en la máquina para emitir los niveles de cada depósito y otro en una hipotética centralita en la que se recibirían dichos datos.

### 8.1. Jetson Nano

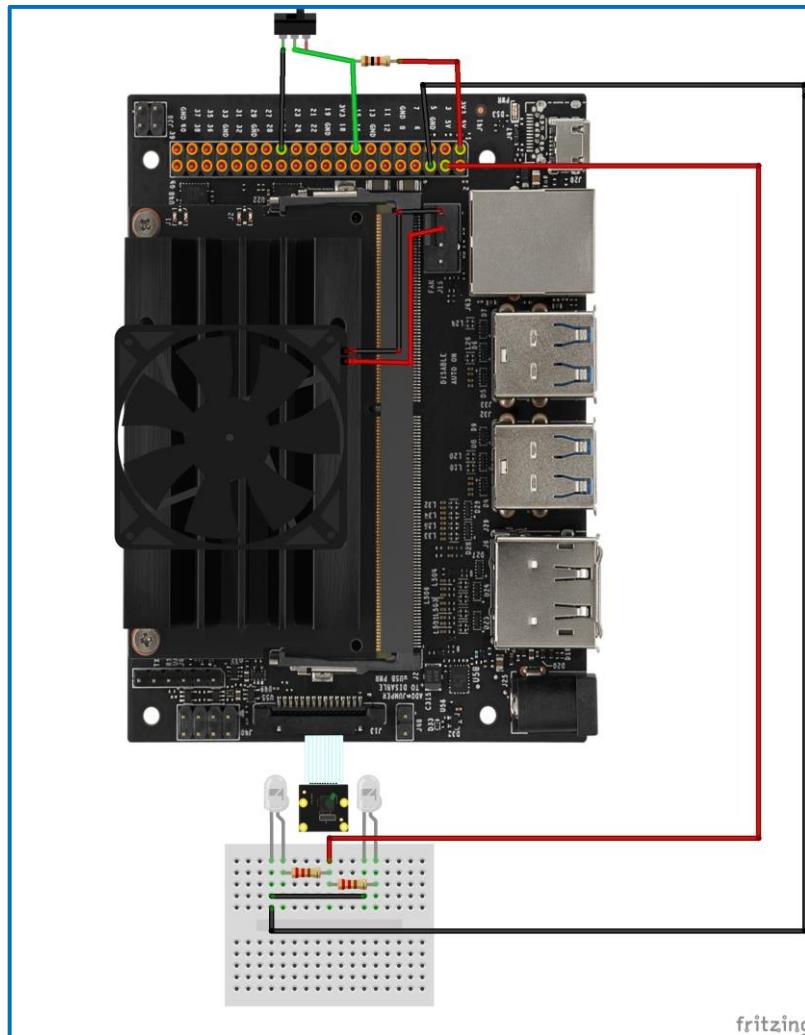
El “cerebro” del proyecto es el microprocesador Jetson Nano de la marca Nvidia®. Se encargará de ejecutar los siguientes subprocessos en diferentes dispositivos:



**Figura 84:** Esquema de conexiones y comunicaciones del Jetson Nano  
**Fuente:** Elaboración propia

- Detección de movimiento y captura de una fotografía para clasificar el residuo. Para ello se conecta mediante un cable flex una cámara **Pi Noir V2**<sup>32</sup> ideal para entornos con baja iluminación como puede ser el interior de esta máquina, no obstante, se equipará con dos LEDs blancos y sus correspondientes resistencias de  $220\Omega$  (*Ilustración 67*).

<sup>32</sup> Junto a la cámara, como se detalló en el epígrafe de diseño, se acoplan dos LEDs blancos para iluminar mejor la escena, se conectan mediante dos resistencias de  $330\Omega$  a uno de los pines de 5V que ofrece la placa Jetson Nano y por el otro extremo a tierra (GND). Por ello, cada vez que se conecte la placa, se encenderán.



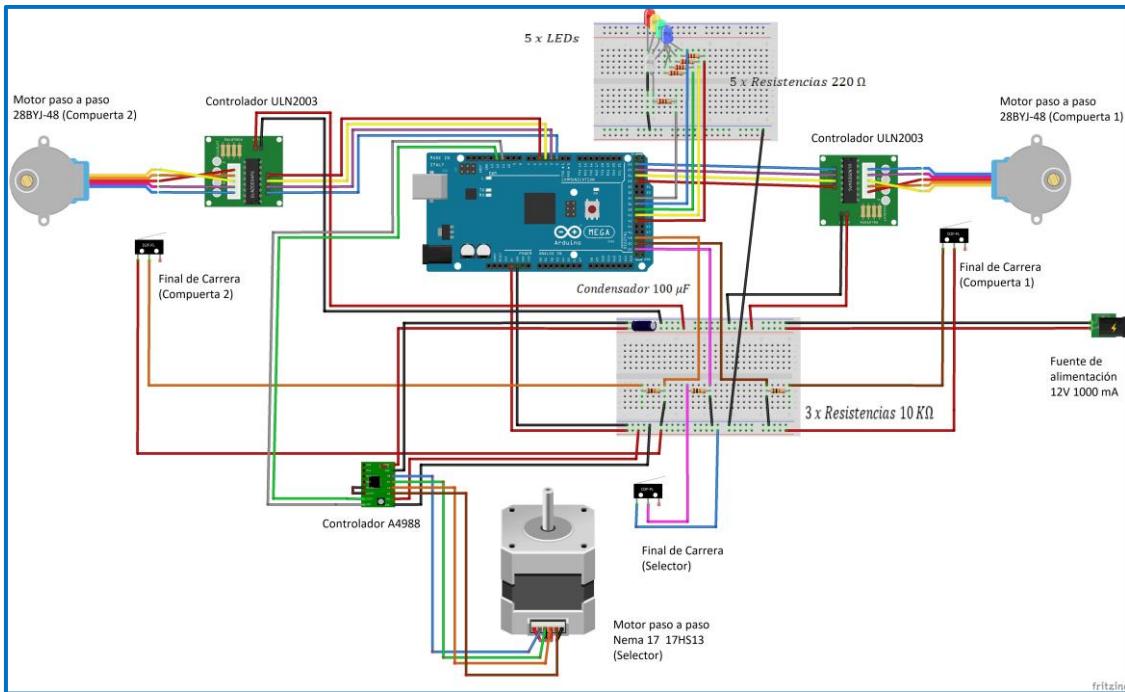
**Figura 85:** Esquema de conexiones del Jetson Nano  
**Fuente:** Elaboración propia

También cabe mencionar que se ha acoplado un ventilador de 5V al radiador para refrigerar la placa de la Jetson Nano, este se conectó a los pines de 5 V y GND del módulo J15. Y un interruptor para iniciar o pausar la ejecución del programa que controla la máquina, que se conectó mediante una resistencia en Pull Up al pin 15.

- Gestión de los movimientos de compuertas y selector. Estas tareas serán ejecutadas por el **Arduino Mega** tras recibir este una orden de la Jetson Nano mediante un bus de tipo USB (Universal Serial Bus USB) conectado al puerto serie del Arduino (UART).
- Comunicación LoRa efectuada por un **ESP32 con modulo LoRa** de la marca TTGO, conectado también mediante el puerto UART.

## 8.2. Arduino Mega

Con la finalidad de liberar de carga de trabajo a la Jetson Nano, un Arduino Mega será el encargado de efectuar los movimientos necesarios para el funcionamiento de la máquina y ejecutar algunos elementos de señalización.



**Figura 86:** Esquema de conexiones del Arduino Mega  
**Fuente:** Elaboración propia

### 8.2.1. Movimiento de compuertas

Para el movimiento de las compuertas se conectarán dos motores 28BIJ-48 a sus respectivos controladores ULN-2003AN acoplados en un PCB, en los que se incluyen cuatro LEDs indicativos de cada paso que da el motor, así como también de sus cuatro resistencias correspondientes, de  $510\ \Omega$ . Estos dos controladores se conectarán a una fuente de tensión de 12 V y 1 000 mA. Los motores de las compuertas funcionarán normalmente a la par, salvo que una compuerta se demore más que la otra, por lo tanto, la intensidad que recibirá cada uno de ellos rondará los 0,5 A, intensidad nominal que establece el fabricante de dichos motores.

Los cuatro pines asociados a las cuatro bobinas de cada motor se conectan al Arduino Mega, de forma que el motor de la compuerta 1 se conecta a los pines 22, 24, 26 y 28. Y el motor de la compuerta opuesta (compuerta 2) se conecta a los pines 2, 3, 4 y 5.

Para poder realizar un *homing*, es decir establecer un origen del movimiento de cada motor. En este caso cuando las compuertas están en su posición superior, cerradas, se emplean dos interruptores con rodillo normalmente abiertos acoplados a la sujeción de las varillas de la parte superior. Estos interruptores se conectan al Arduino Mega mediante un circuito de tipo Pull Down, con resistencias de  $10\ K\Omega$ , a los pines de 5V, GND y en el caso de la compuerta 1, al pin 50 y la compuerta 2, al pin 48.

### 8.2.2. Movimiento del selector

Para realizar la rotación del selector se ha empleado un motor “paso a paso” NEMA 17HS13, porque garantizaba el torque necesario para efectuar tal movimiento, aproximadamente, unos 2,5 kgf.cm frente a, apenas, 0,5 kgf.cm que ofrecen los 28BYJ-48.

Este motor necesita de un controlador A4988 que se conectará a la misma fuente de tensión de 12 V anteriormente mencionada, poniendo en paralelo un condensador de  $100 \mu F$  para garantizar una tensión mantenida y constante al motor. Se puede emplear la misma fuente de tensión porque este movimiento no coincidirá nunca con el de las compuertas.

Los pines necesarios para controlar este motor son: uno que determina el sentido de giro del motor conectado al pin 11 del Arduino y otro para ejecutar un paso, conectado al pin 12.

También es necesario conectar el pin VCC del A4988 a los 5 V del Arduino y la tierra GND a la tierra del Arduino para la parte lógica del circuito.

Por último, es necesario conectar el pin de Reset del A4988 al Sleep del mismo dispositivo.

Para poder establecer el origen del movimiento del selector se acopla un interruptor, normalmente abierto a la pieza intermedia, que conecta la máquina con el tubo selector mediante la pieza “sujeción final de carrera selector”. Este interruptor es accionado por el engranaje conducido al ser este desplazado. Se conecta una vez más mediante un circuito Pull Down y una resistencia de  $10 K\Omega$  a 5V, GND y al pin 52 del Arduino.

### **8.2.3. Señalización del residuo a procesar**

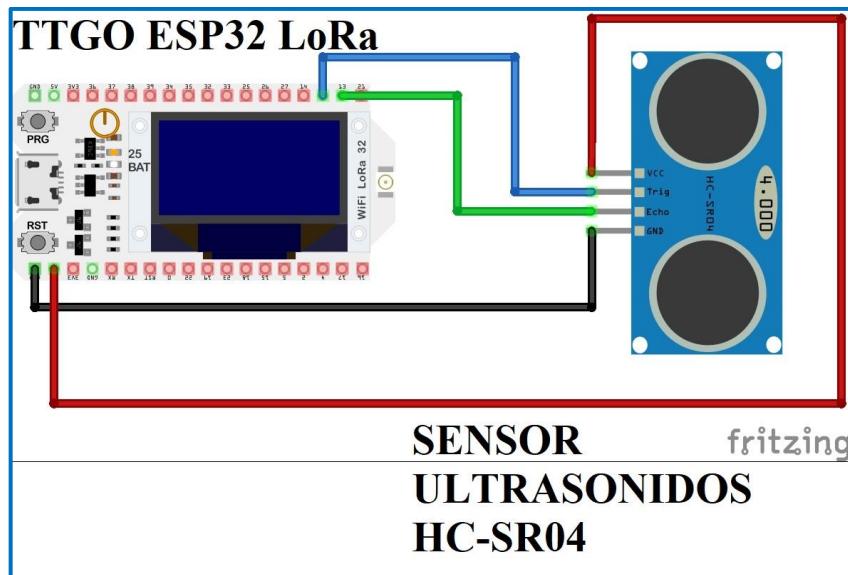
Para poder conocer el resultado del proceso de clasificación de una forma inmediata, se conectan cinco LEDs mediante resistencias de  $220 \Omega$  a los siguientes pines:

- El LED rojo para el metal al pin 42
- El LED verde para el vidrio al pin 38
- El LED azul para el cartón/papel al pin 36
- El LED amarillo para el plástico al pin 40
- El LED blanco para el orgánico u otro tipo de residuo no reconocible al pin 34

## **8.3. ESP32 LoRa**

Este dispositivo se encargará de que cuando se deposite un residuo en uno de los contenedores, es decir tras recibir una orden del Jetson Nano, ejecute una medición mediante un sensor de ultrasonido HC-SR04 acoplado en la parte inferior del tubo selector. El pin ECHO de este sensor se conectará al pin 13 del ESP32, el TRIGGER al pin 12, el VCC al pin de 5V y el GND al pin del mismo nombre del dispositivo.

Para el correcto funcionamiento del LoRa es necesario conectar la antena de 900 MHz que incluye el módulo del ESP32.

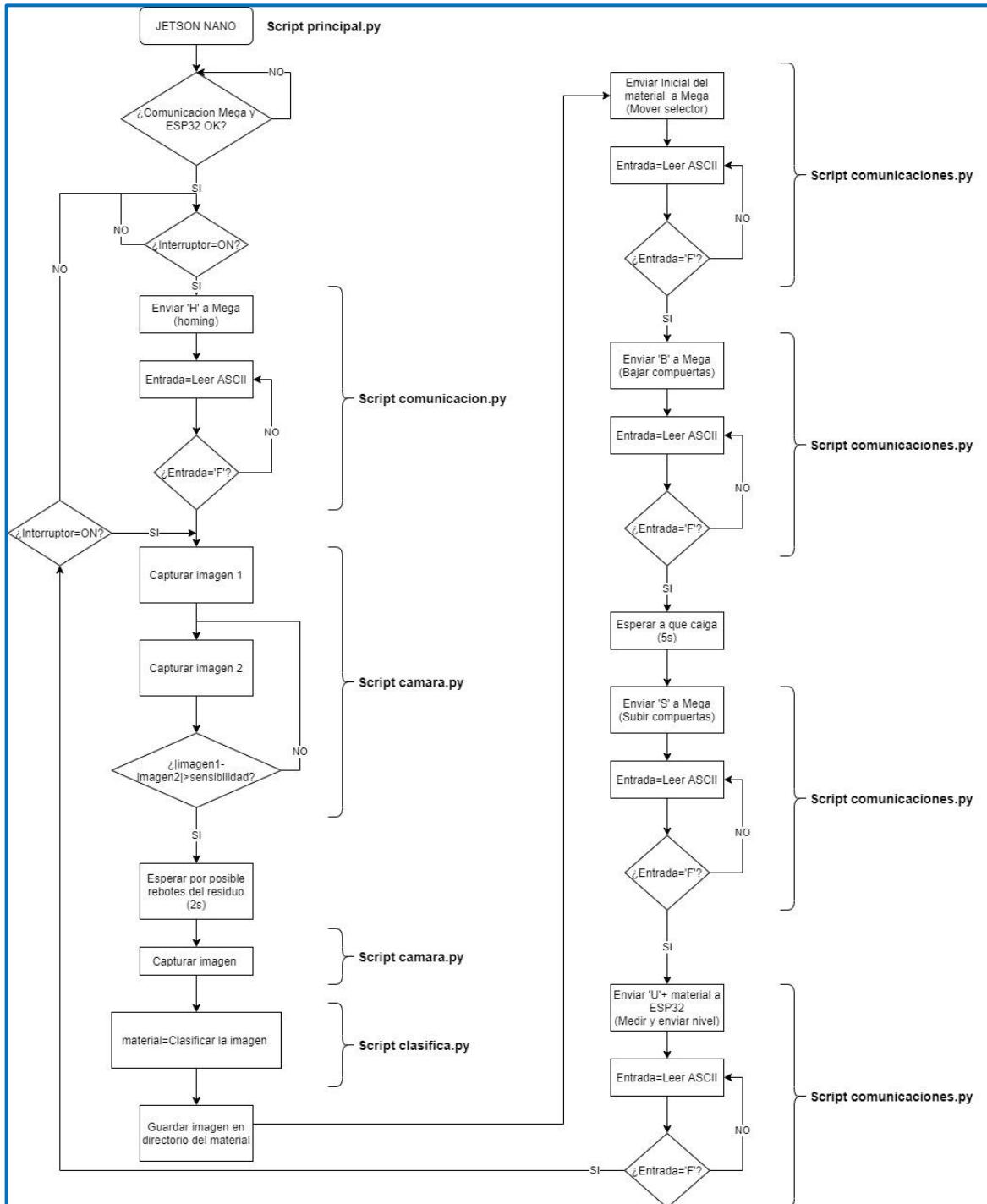


**Figura 87:** Esquema de conexiones del ESP32

Fuente: Elaboración propia

## 9. PROGRAMACIÓN

### 9.1. Jetson Nano



**Figura 88:** Diagrama de flujo del programa principal del Jetson Nano  
**Fuente:** Elaboración propia

Como se puede ver en la *Figura 88*, la programación en el Jetson Nano se descompuso en diferentes *scripts* programados en lenguaje Python.

El *script* principal (*principal.py*) ejecuta un *homing* de los mecanismos de compuertas y del selector, una vez el interruptor conectado al Jetson Nano (epígrafe 8.1) es activado. Para

esto envía en ASCII el carácter “H” al Arduino Mega, para ello recurre a otro *script* (comunicación.py) encargado de las comunicaciones ASCII tanto con el Arduino como con el ESP32 encargado de las comunicaciones LoRa y la medición de los niveles.

Una vez finalizado el *homing*, al recibir una “F” por parte del Arduino, ejecuta un método del *script* encargado de manejar la cámara (cámara.py) que detecta movimiento. Para ello, capta una primera imagen y la compara con sucesivas imágenes hasta que exista una que difiera de la primera en una cantidad determinada de píxeles (sensibilidad). En ese caso volverá al programa principal, esperará dos segundos para que el residuo termine de asentarse y le tomará una imagen.

La ruta de esta imagen será enviada por parámetro a un cuarto *script* (clasificación.py) que mediante el método “clasificar”, realiza la inferencia<sup>33</sup> sobre la imagen tomada, devolviendo al programa principal una variable de tipo *string* que determina el residuo al que pertenece.

Sabiendo esto se guardará dicha imagen en un directorio de su clase, con el nombre de su tipo de residuo más un identificador que se autoincrementa con cada residuo de su clase procesado. Por ejemplo, el segundo plástico en ser procesado se guardaría con la siguiente ruta “residuos\_procesados/plástico/plástico2.jpg”. Con esto se genera una base de datos con imágenes de residuos procesados.

A continuación, se envía al Arduino el residuo que se debe de procesar, es decir, se envía su inicial en ASCII, para que este desplace el selector hasta la posición correcta.

Luego se envía una “B” para hacer bajar las compuertas, se esperan cinco segundos a que el residuo caiga por completo y se envía una “S” para hacer subir de nuevo las compuertas. Cuando recibe la última “F” indicando que el Arduino ha finalizado, se envía al ESP32 una “U” acompañado de un número del 0 al 4 que determina el residuo clasificado. Se envía en formato numérico para facilitar el procesamiento por parte del ESP32.

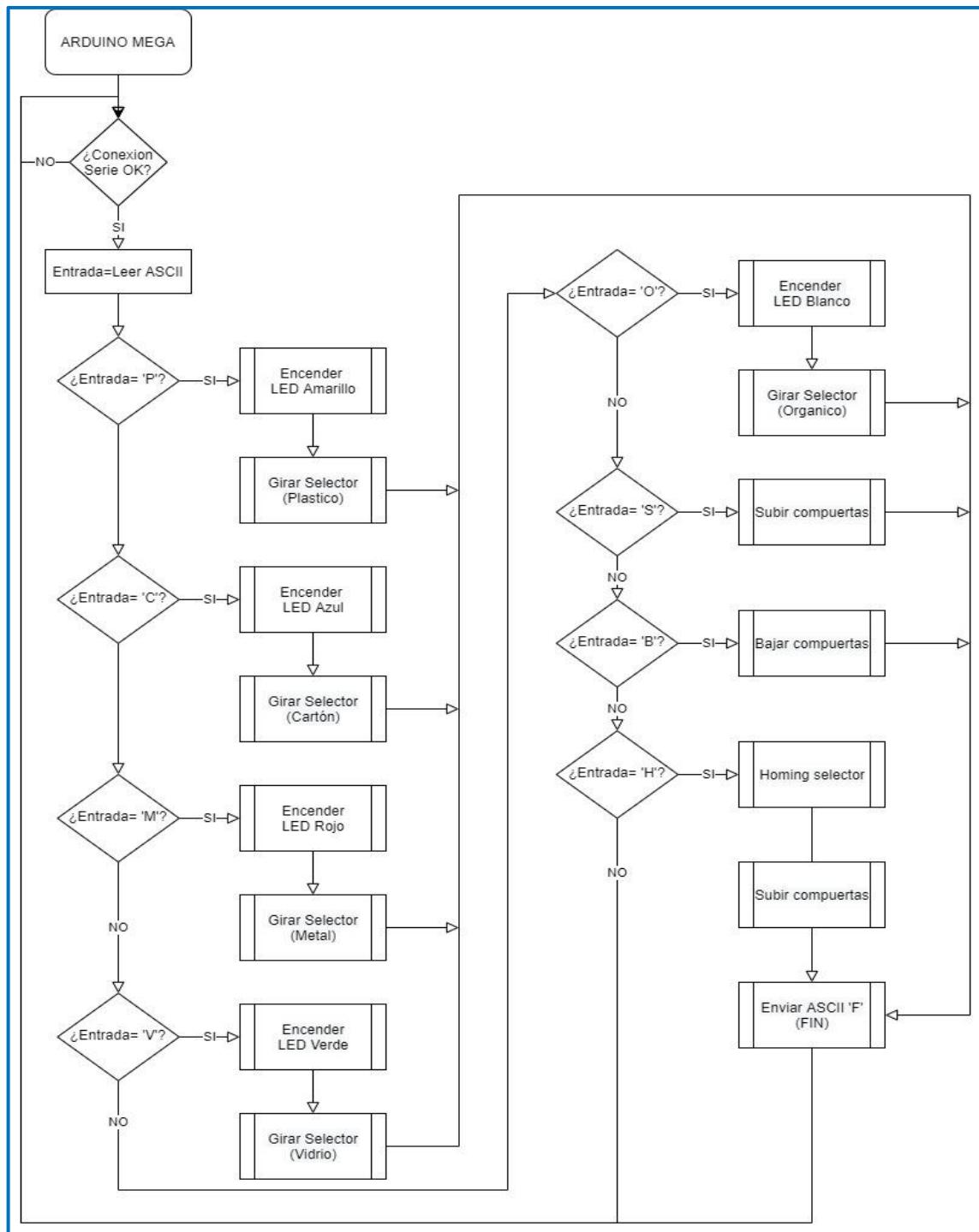
Una vez el ESP32 indica que ya ha realizado la medición y la transmisión de los datos, se vuelve al estado de “esperar movimiento” por si un nuevo residuo vuelve a ser depositado. De esta forma se podrá ejecutar el mismo proceso indefinidamente, a no ser que se desactive el interruptor, en ese caso finalizará el procesamiento del residuo que se esté procesando si es que existe y la maquina se quedaría a la espera de una nueva activación que, de realizarse, conllevaría volver a realizar un nuevo *homing*.

## 9.2. Arduino Mega

El Arduino está constantemente comprobando si recibe algún carácter codificado en ASCII por parte de la Jetson Nano para ejecutar los subprocesos que se reflejan en la *Figura 89* siguiente.

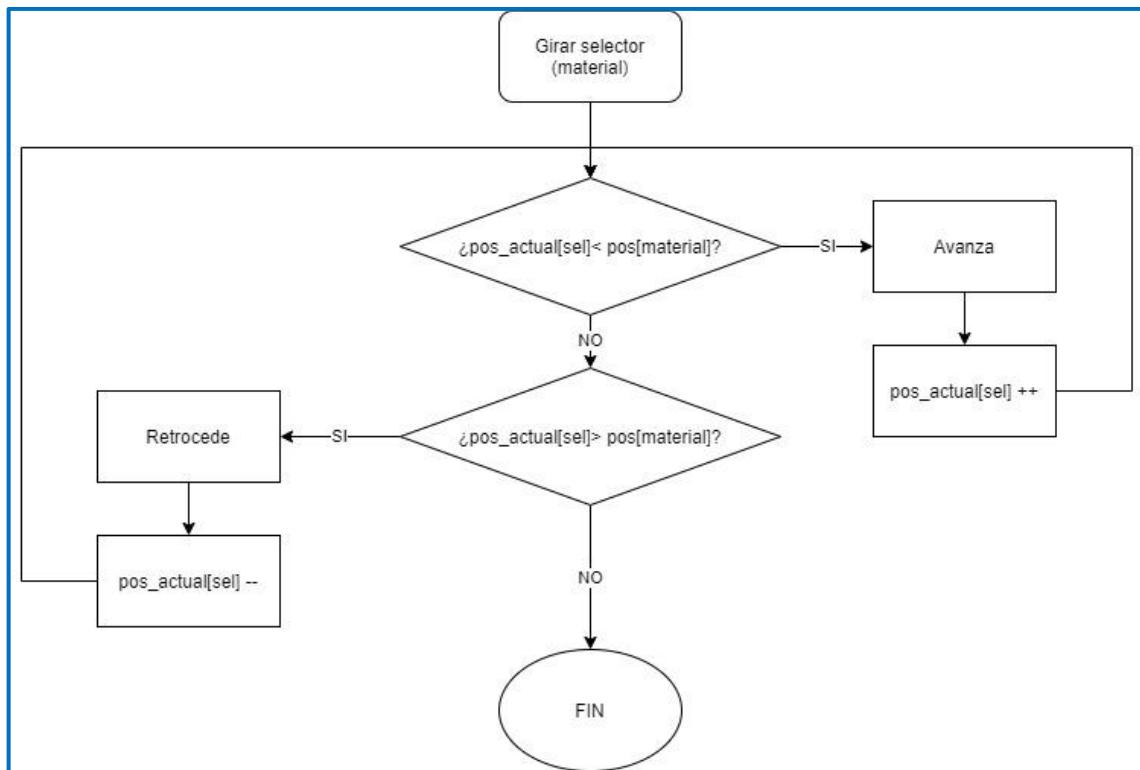
---

<sup>33</sup> Descrita en el epígrafe 5.5



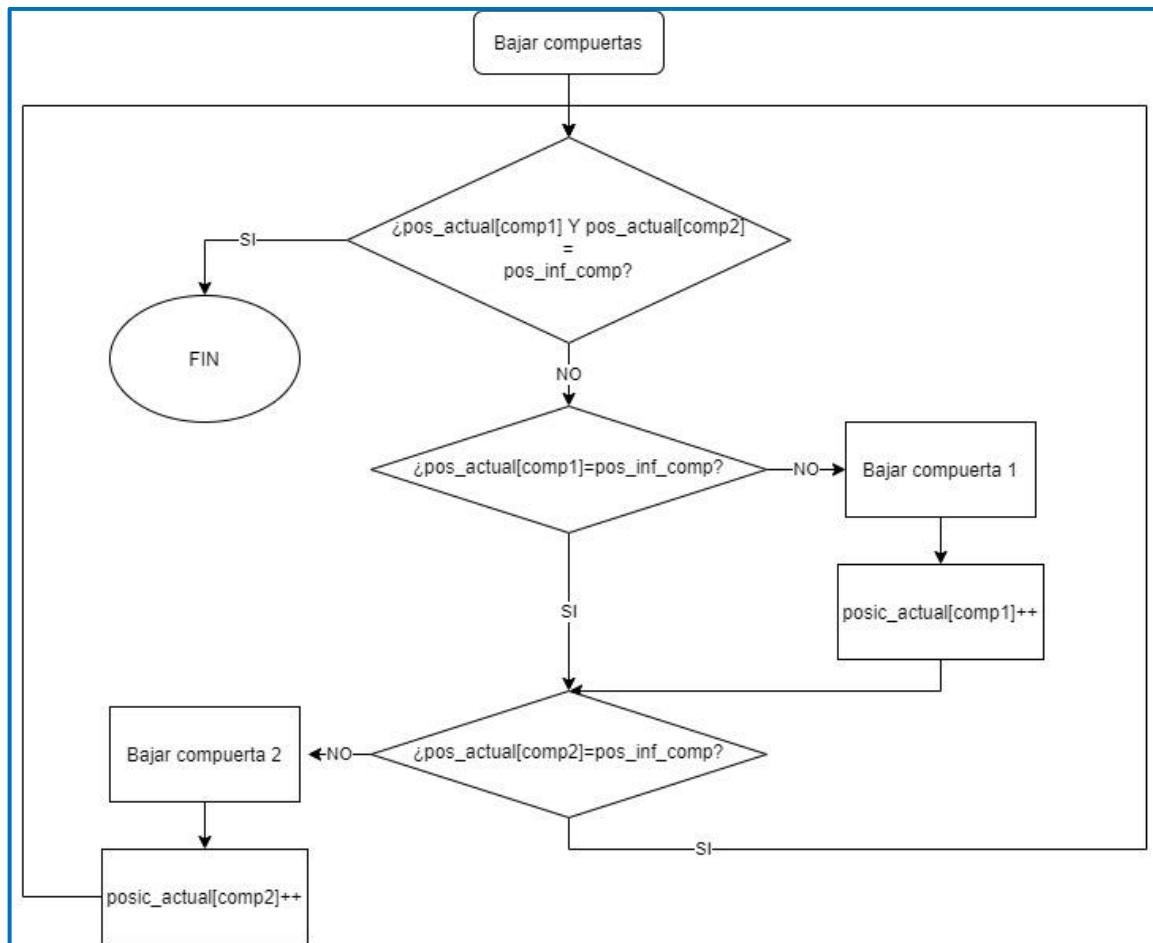
**Figura 89:** Diagrama de flujo del programa del Arduino Mega  
**Fuente:** Elaboración propia

Si recibe la inicial de algún tipo de residuo, enciende el LED correspondiente y desplaza el selector hasta el depósito correcto ejecutando el siguiente subproceso.



**Figura 90:** Diagrama de flujo del proceso para girar el selector  
**Fuente:** Elaboración propia

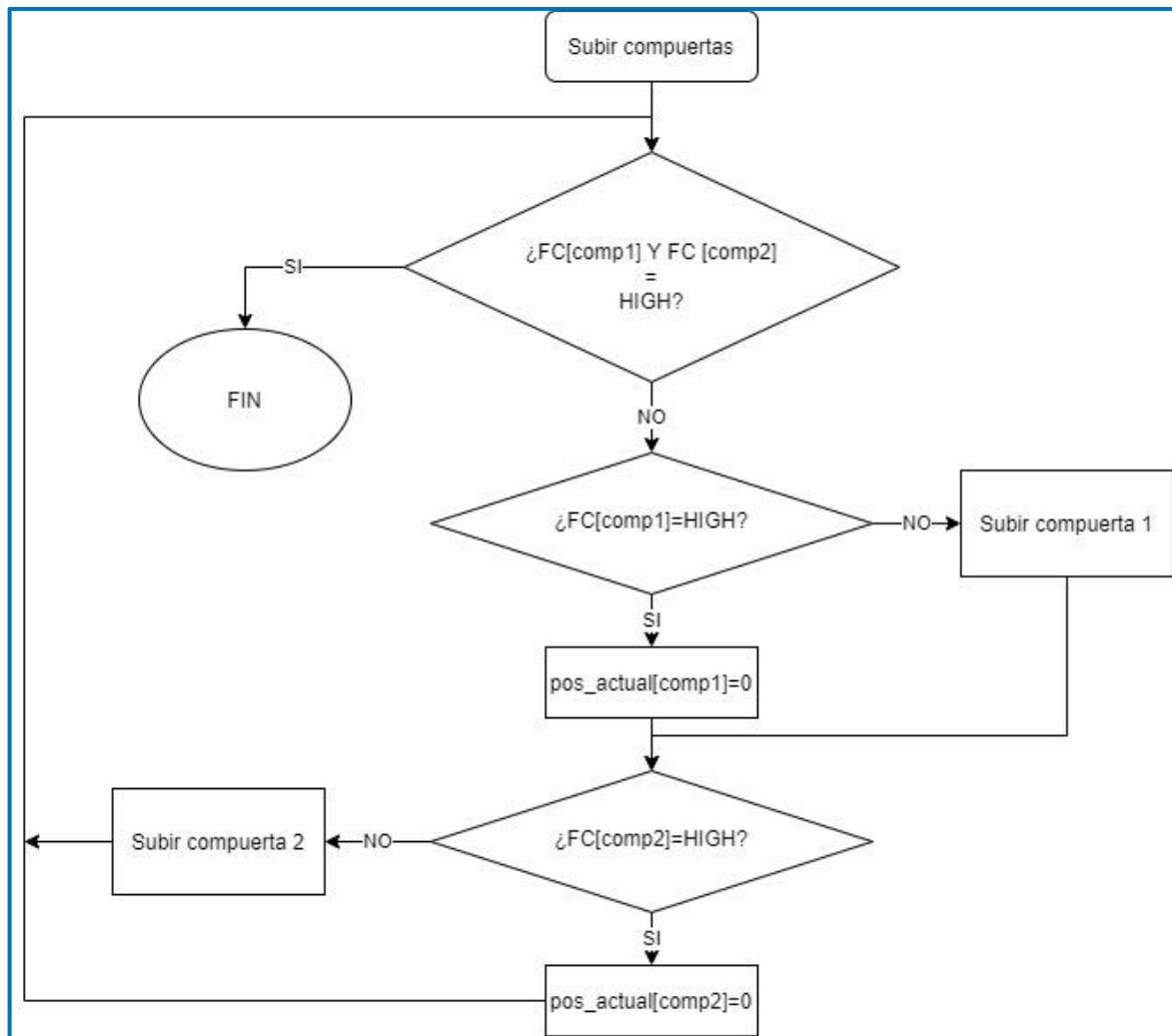
Si la posición del selector, referenciada a un origen al haberse realizado previamente un proceso de *homing*, es menor que la posición objetivo del depósito en cuestión, el motor se desplaza en sentido de avance e incrementa por tanto la variable de la posición actual del selector ('*pos\_actual[sel]*'). De lo contrario, retrocede y decrementa dicha variable.



**Figura 91:** Diagrama de flujo del proceso para bajar las compuertas

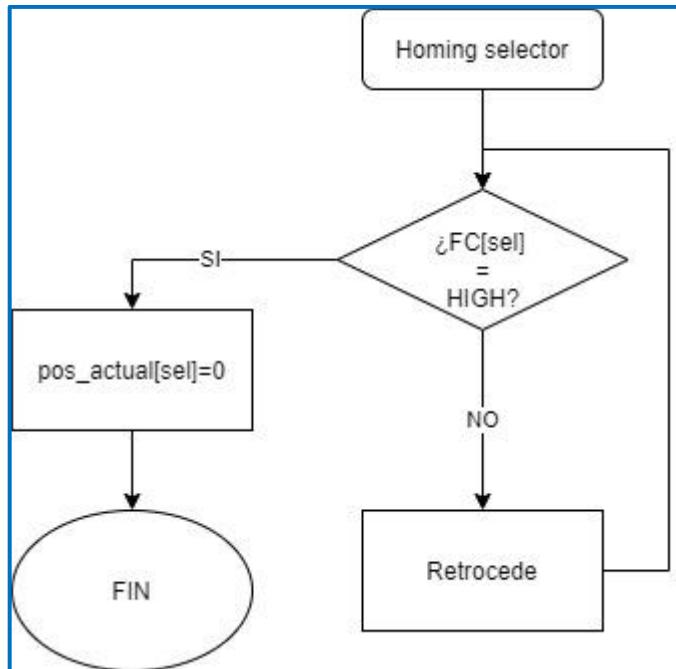
**Fuente:** Elaboración propia

Si recibe una “B”, hace bajar las compuertas moviendo el motor de cada compuerta de forma independiente (*Figura 91*). De tal manera que mientras las dos compuertas no hayan alcanzado la posición que se considera como que se encuentran bajadas por completo (*pos\_inf\_comp*), se ejecutarán los pasos en cada motor y se incrementará la variable de posición de cada compuerta (‘*pos\_actual[compX ]*’).



**Figura 92:** Diagrama de flujo del proceso para subir las compuertas  
**Fuente:** Elaboración propia

Si recibe una “S” sube las compuertas de forma independiente hasta que las dos hayan accionado sus respectivos finales de carrera (*Figura 92*).



**Figura 93:** Diagrama de flujo del proceso para realizar el homing del selector

**Fuente:** Elaboración propia

Por último, si recibe una ‘H’ ejecuta el proceso de *homing* del selector desplazandolo hasta el final de carrera y actualizando la variable ‘pos\_actual[sel]’ a 0 cuando llegue a él.

Seguidamente, realiza el mismo proceso de *homing* con las compuertas ejecutando el subproceso de subir compuertas anteriormente comentado.

## 10. COMUNICACIÓN DE LOS DATOS MEDIANTE LoRa

Cada vez que un residuo se deposite en su contenedor correspondiente, el ESP32 medirá, con un sensor de ultrasonido, el nivel del depósito. Esta información será emitida de inmediato a través de comunicación LoRa a otro ESP32 actuando como receptor desde una hipotética centralita. Este receptor, a su vez, se conectará a una red WiFi™ y generará un servidor web al que se accederá desde una aplicación móvil introduciendo previamente la IP asignada al ESP32 por parte del router.



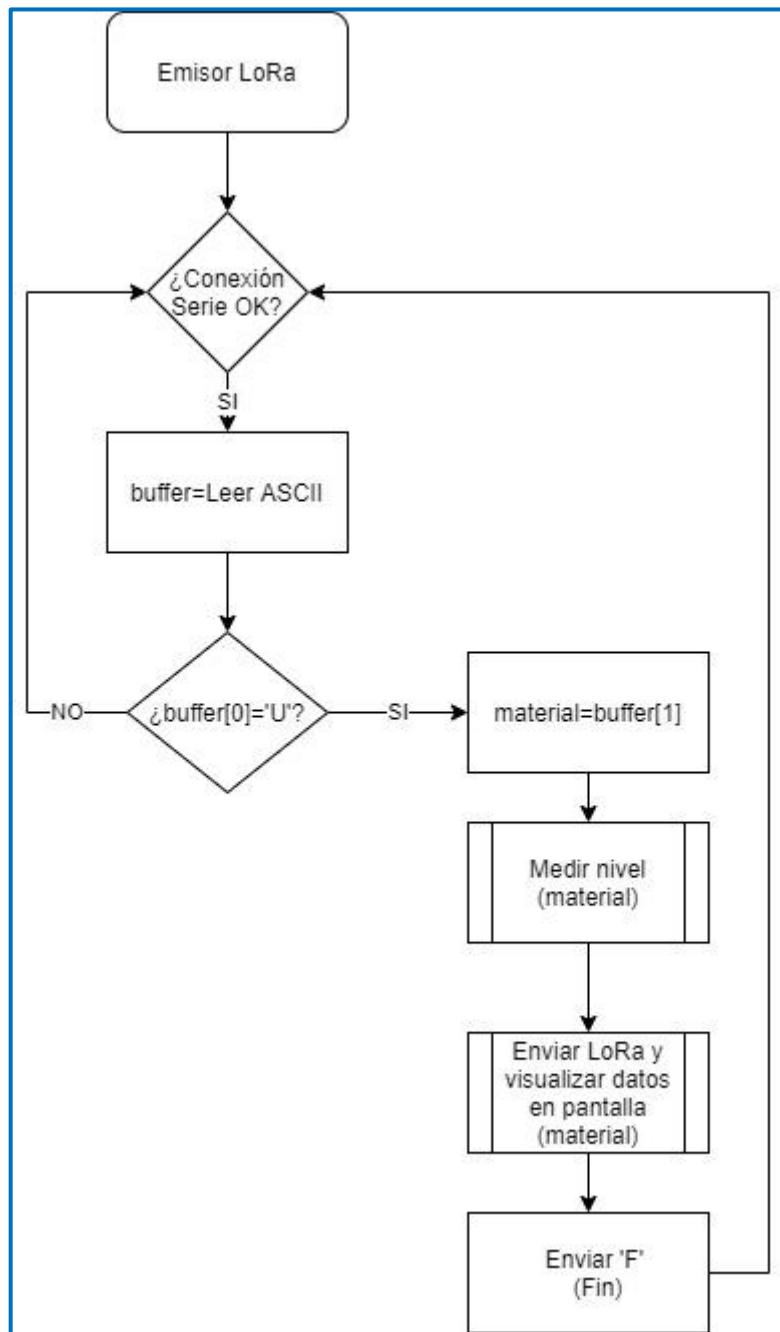
**Figura 94:** Esquema del proceso de captación de los niveles, transmisión y visualización  
**Fuente:** Elaboración propia

LoRa es una comunicación dentro de las redes de tipo LPWAN (*Low-Power Wide Area Network*) patentada por la compañía Semtech. Está comenzando a ser muy empleada en dispositivos y proyectos relacionados con el IoT (*Internet of Things*) integrado incluso en los ecosistemas de las ciudades inteligentes o *Smart City*. Esto se debe a que son dispositivos de muy bajo consumo, es decir, no precisan de conexión a la red eléctrica, simplemente son necesarias baterías o paneles solares para operar durante largos períodos de hasta, incluso, años. También por su extenso rango, en el caso del LoRa puede alcanzar los 20 km (en el caso del módulo empleado, 5 km). Y, en contraposición a otras alternativas como módulos LTE, no requieren un intermediario como una operadora de telefonía móvil, con el abaratamiento y libertad que ello conlleva.

En este epígrafe se detallará el código de programación, ya que no es sencillo entender los programas solo mediante diagramas de flujo.

Fue desarrollado en Arduino y se basó, en algunos aspectos, en el proyecto de Rui Santos “ESP32 LoRa Sensor Monitoring with Web Server” (24) con ciertas modificaciones para adaptarlo al proyecto en cuestión y a la comunicación ASCII con la que se gestionan los subprocesos como este.

## 10.1. Emisor LoRa<sup>34</sup>



**Figura 95:** Diagrama de flujo del programa del emisor LoRa  
**Fuente:** Elaboración propia

Para comenzar con la secuencia de medida y envío de los datos, el ESP32 recibirá del Arduino Mega mediante el puerto USB en carácter ASCII la letra “U” seguida del tipo de residuo al que pertenece, de forma que si recibe un 0 será plástico; 1, cartón; 2, metal; 3, vidrio; y 4, orgánico. Esto provocará que realice la medida del depósito correspondiente, actualice solo la variable del residuo en cuestión y envíe, bajo el protocolo LoRaWan™, solo el nivel modificado al receptor, además de un ID que se autoincrementa con cada paquete de datos enviado.

<sup>34</sup> Código completo del programa en el Anexo 15.2.1 Código de programa

### 10.1.1. Importación de librerías

```
15 #include <Arduino.h>
16
17 //Librerias para la gestión del LoRa
18 #include <SPI.h>
19 #include <LoRa.h>
20
21 //Librerias para la visualización en la OLED
22 #include <Wire.h>
23 #include <Adafruit_GFX.h>
24 #include <Adafruit_SSD1306.h>
```

Figura 96: Emisor LoRa. Importación de librerías

Fuente: Elaboración propia

Se comienza, como cualquier código, importando las librerías. En este caso, al estar programando en *Visual Studio Code* con la extensión de PlatformIO, es necesario importar la librería Arduino.

Para la comunicación LoRa son necesarias las librerías *SPI.h* y *LoRa.h*. Para la visualización en la OLED integrada del ESP32 son necesarias las librerías *Wire.h*, *Adafruit\_GFX.h* y *Adafruit\_SSD1306.h*.

### 10.1.2. Definición de variables

```
27 //se definen los pines para el modulo LoRa que incorpora el TTGO
28 #define SCK 5 //señal de reloj
29 #define MISO 19 //Master Input Slave Output señal de entrada
30 #define MOSI 27 //Master Output Slave Input señal de salida
31 #define SS 18 //Slave Select selecciona el esclavo al que se envían
32 #define RST 14 //boton de reset
33 #define DIO0 26 //
34
35 #define BAND 866E6 //se definen la banda de transmisión en europa es la 866E6
36
37 //Se definen los pines de la OLED
38 #define OLED_SDA 4
39 #define OLED_SCL 15
40 #define OLED_RST 16
41 #define SCREEN_WIDTH 128 // OLED ancho de la pantalla en pixeles
42 #define SCREEN_HEIGHT 64 // OLED alto de la pantalla en pixeles
```

Figura 97: Emisor LoRa. Definición de variables

Fuente: Elaboración propia

En el ESP32, el módulo LoRa está conectado a la placa de la siguiente forma: la señal de reloj (SCK) está conectada al pin 5; la señal de datos de entrada o *Master Input Slave Output* (MISO), en el pin 19; la señal de datos de salida o *Master Output Slave Output* (MOSI), en el pin 27; la señal que selecciona el esclavo destinatario de los datos o *Slave Select* (SS) está conectada en el pin 18; y el Reset (RST), en el 14.

Es necesario también seleccionar la banda de transmisión europea (868MHz) para cumplir con la normativa vigente.

Por último, es preciso indicar a que pines se conecta la OLED que ya está incorporada en el módulo de TTGO.

```
44 int readingID = 0; //variable para el ID del mensaje
45
46 int counter = 0; //contador
47 String LoRaMessage = ""; //en esta string se irá creando el mensaje para enviar
48
49 // vector donde se guardarán los niveles de los 5 depósitos
50 float nivel[5]={0,0,0,0,0} ; //0->Plástico ; 1->Papel/Cartón ; 2->Metal ;3-> Vidrio
   ;4->Orgánico
51 int material ;
52 const int EchoPin = 13; // el pin ECHO del sensor de ultrasonidos se conecta al pin 13
53 const int TriggerPin = 12; //el trigger se conecta al pin 12
54
55 //se crea un objeto display de tipo Adafruit_SSD1306 con las características de la pantalla
56 Adafruit_SSD1306 display(SCREEN_WIDTH, SCREEN_HEIGHT, &Wire, OLED_RST);
```

**Figura 98:** Emisor LoRa. Definición de variables

**Fuente:** Elaboración propia

Se declara una variable *readingID* de tipo *int* para guardar el número de paquete enviado; un contador que más adelante se comentará; y una *string* (LoRaMessage) en la que se irá montando el mensaje a enviar mediante LoRa.

Así mismo, se declara un vector de tipo *float* denominada “nivel”, en la que se irán guardando los valores de nivel de cada depósito, siendo la posición 0 de este vector el depósito del plástico; la 1º, del papel y cartón; la 2º, del metal; la 3º, del vidrio; y la 4º del orgánico.

Por tanto, será necesario también una variable de tipo *int* (“material”) que indique la posición en dicho vector del residuo que se está clasificando en ese momento, para poder actualizar su valor.

El pin del Echo se conectará al pin 13 del ESP32 y el del *Trigger*, al pin 12.

Por último, se crea un display de tipo Adafruit\_SSD1306 con las características de la pantalla.

### 10.1.3. Función de inicialización de la OLED

```

59 //Inicializa la pantalla OLED
60 void startOLED(){
61     //resetea el OLED
62     pinMode(OLED_RST, OUTPUT);
63     digitalWrite(OLED_RST, LOW);
64     delay(20);
65     digitalWrite(OLED_RST, HIGH);
66
67     //inicializa el OLED
68     Wire.begin(OLED_SDA, OLED_SCL);
69     if(!display.begin(SSD1306_SWITCHCAPVCC, 0x3c, false, false)) { // Address 0x3C for
128x32
70         Serial.println(F("SSD1306 allocation failed"));
71         for(;); // Don't proceed, loop forever
72     }
73     display.clearDisplay(); //limpia la pantalla
74     display.setTextColor(WHITE); //establece como blanco el color de escritura
75     display.setTextSize(1); //establece el tamaño del texto en una fila
76     display.setCursor(0,0); //pone el cursor en el origen (parte superior izquierda)
77     display.print("Emisor LoRa");
78 }
```

**Figura 99:** Emisor LoRa. Inicialización de la OLED

**Fuente:** Elaboración propia

Es necesario inicializar la pantalla, es decir, se resetea; se comprueba el correcto funcionamiento; se limpia; se establece el color y tamaño del texto; se coloca el cursor en el origen; y, por último, se escribe en la pantalla el nombre del dispositivo, en este caso “Emisor LoRa”.

### 10.1.4. Función de inicialización del LoRa

```

80 //Inicializa el modulo LoRa
81 void startLoRA(){
82     int counter;
83
84     SPI.begin(SCK, MISO, MOSI, SS); //establece los pines correspondientes
85     LoRa.setPins(SS, RST, DIO0); //establece el modulo LoRa
86
87 //hace una prueba para comprobar si se establece conexión en un tiempo determinado
88     while (!LoRa.begin(BAND) && counter < 10) { //envía hasta 10 paquetes mientras no
haya establecido conexión
89         Serial.print(".");
90         counter++;
91         delay(500);
92     }
93     if (counter == 10) { //si se han enviado los 10 paquetes quiere decir que se agoto
el tiempo de conexión
94         Serial.println("Error: fallo de inicializacion"); //salta un error
95     }
96     Serial.println("Inicializa LoRa OK!"); //si se establecio conexión en la pantalla se
muestra este mensaje
97     display.setCursor(0,10); //desplaza el cursor 10 columnas a la derecha
98     display.clearDisplay(); //limpia la pantalla
99     display.display(); //lo muestra
100    delay(2000);
```

**Figura 100:** Emisor LoRa. Función de inicialización del LoRa

**Fuente:** Elaboración propia

Para inicializar el LoRa se ejecutará esta función que establece los pines involucrados anteriormente comentados. Posteriormente, realiza una prueba enviando paquetes mientras no haya establecido conexión con el emisor. Si se han enviado 10 paquetes quiere decir que se ha

agotado el tiempo de conexión y, por tanto, señaliza un error de inicialización. De lo contrario, muestra un mensaje de inicialización correcta.

### 10.1.5. Función de medición del nivel

```
//función que devuelve la distancia en cm proporcionada por el sensor de ultrasonidos
int medir (int TriggerPin, int EchoPin) {
    long duracion, distancia_Cm;

    digitalWrite(TriggerPin, LOW); //para generar un pulso limpio pone a LOW 4us
    delayMicroseconds(4);
    digitalWrite(TriggerPin, HIGH); //genera Trigger (disparo) de 10us
    delayMicroseconds(10);
    digitalWrite(TriggerPin, LOW);

    duracion = pulseIn(EchoPin, HIGH); //mide el tiempo entre pulsos, en
    en microsegundos

    distancia_Cm = duracion * 343 * 100 * (1/1000000) / 2; //convierte a distancia,
    en cm
    return distancia_Cm;
}
```

**Figura 101:** Emisor LoRa. Función de medición del nivel

**Fuente:** Elaboración propia

La función “medir” permite obtener la distancia del ultrasonido a los residuos depositados en el cada contenedor. Para ello, genera un disparo en el pin *trigger* de 10 microsegundos provocando la emisión de una onda de ultrasonidos y, mediante la función *pulseIn*, cuenta el tiempo transcurrido hasta recibir el eco de la señal emitida a través del pin Echo en microsegundos. Este tiempo es después convertido a distancia en cm aplicando la siguiente fórmula:

**Ecuación 27:** Transformación del tiempo de propagación de la onda del sensor de ultrasonido a distancia

$$distancia(cm) = \frac{duracion(\mu s) * vel. sonido \left( \frac{m}{s} \right) * \frac{100\ cm}{1\ m} * \frac{1\ s}{10^6\ \mu s}}{2}$$

La distancia al objeto es igual a la duración de mitad del camino que ha realizado la onda entre su emisión y su recepción por la velocidad del sonido (aproximada a 343 m/s).

### 10.1.6. Función de actualización de variables

```
119 // funcion que según el material que se este procesando guarda el nivel medido por
el ultrasonidos en su variable correspondiente
120 void getReadings(int material){
121     int cm = medir (TriggerPin, EchoPin); //obtiene la distancia en cm
122     Serial.print("Distancia: ");
123     Serial.println(cm);
124     float nivel_medido = (-2.56)*cm+ (112.82); //nivel maximo 100 % cuando la
distancia es 5 cm ; nivel minimo 0% cuando distancia es 44 cm
125     nivel[material]= nivel_medido;//el nivel medido se guarda en la variable del
nivel que se este clasificando
126     Serial.print("Nivel: ");
127     Serial.println(nivel_medido);
128 }
```

**Figura 102:** Emisor LoRa. Función de actualización de variables

**Fuente:** Elaboración propia

La función *getReadings* ejecuta la medición de la distancia mediante el ultrasonido, posteriormente transforma esa distancia en cm a un porcentaje en el cual el nivel máximo (100%) se daría cuando la distancia al residuo fuese de 5 cm y el nivel mínimo cuando la distancia a los residuos que hubiese en el depósito fuese de más de 44 cm (aproximadamente la distancia que hay entre el ultrasonido y el fondo del contenedor).

A continuación, este nivel se introduciría en la posición del vector “nivel” correspondiente, gracias a la variable “material” pasada por parámetro.

#### 10.1.7. Función para emitir el paquete de datos mediante LoRa

```
//función que crea el mensaje y lo envia mediante LoRa tambien visualiza la info en la pantalla
void sendReadings(int material) {
    //se crea el mensaje intercalando los datos con marcadores
    LoRaMessage = String(readingID) + "/" + String(material) + '&' +
String(nivel[material]);
    // se envia el paquete
    LoRa.beginPacket();
    LoRa.print(LoRaMessage);
    LoRa.endPacket();

    display.clearDisplay(); //se limpia la OLED
    display.setCursor(0,0); //se desplaza el cursor al origen
    display.setTextSize(1); //se establece el tamaño de letra en 1 fila y 1 columna
    display.print("Paquete " + String(readingID) + " enviado"); //se escribe
    display.setCursor(0,10);
    display.print("Plastico:"); //se escribe
    display.setCursor(72,10); //se desplaza el cursor 72 pixeles a la derecha
    display.print(nivel[0] ); //se escribe el nivel de plastico
    display.setCursor(0,20); //se desplaza el cursor 10 pixeles más abajo
    display.print("Carton:");
    display.setCursor(54,20);
    display.print(nivel[1]);
    display.setCursor(0,30);
    display.print("Metal:");
    display.setCursor(54,30);
    display.print(nivel[2]);
    display.setCursor(0,40);
    display.print("Vidrio:");
    display.setCursor(54,40);
    display.print(nivel[3]);
    display.setCursor(0,50);
    display.print("Organico:");
    display.setCursor(54,50);
    display.print(nivel[4]);
    display.display();
    Serial.print("Sending packet: ");
    Serial.println(readingID); //se muestra el ID tambien a traves del monitor serial
    readingID++;
}
```

Figura 103: Emisor LoRa. Función para emitir el paquete de datos mediante LoRa

Fuente: Elaboración propia

Mediante la función *sendReadings*, y pasándole por parámetro el material que ha sido depositado y por tanto que debe de ser actualizado su nivel, genera y emite el mensaje necesario y visualiza toda la información a través del display. Para ello, primero se genera el cuerpo del

mensaje a emitir, en dicho mensaje se incorpora el ID del paquete que se va a enviar (*readingID*). A continuación, se añade un marcador con un carácter ASCII aleatorio como ‘/’ para separar los datos y facilitar su posterior lectura por parte del receptor, la posición del material clasificándose dentro del vector *nivel*, otro marcador aleatorio ‘&’ y, finalmente, el nivel del depósito correspondiente (*nivel[material]*).

Una vez formado el mensaje se envía mediante LoRa con las funciones *Lora.beginPacket()* para iniciar el paquete de datos, *LoRa.print* para escribir el mensaje y *LoRa.endPacket()* para finalizar el mensaje.

Posteriormente se visualizan los datos en la OLED. Para ello se limpia la pantalla; se desplaza el cursor al origen; se establece un tamaño de letra de una fila y una columna; y se escribe el número de paquete enviado seguido por los niveles de cada deposito. Por último, se señaliza, mediante el monitor serial, el paquete enviado y se autoincrementa en una unidad la variable *readingID*.

#### 10.1.8. Función Setup

```
170 void setup() {  
171  
172     Serial.begin(9600); //inicializa la comunicacion serial a 9600 baudios  
173     while (!Serial) {  
174         ; // espera hasta que la comunicación serial se establezca  
175     }  
176     pinMode(TriggerPin, OUTPUT); //se configura el pin del trigger como salida  
177     pinMode(EchoPin, INPUT); //se configura el pin del echo como entrada  
178  
179     startOLED(); //se llama a la función para inicializar la OLED  
180     startLoRA(); //se llama a la función para inicializar EL LORA  
181 }
```

Figura 104: Emisor LoRa. Función *Setup*

Fuente: Elaboración propia

Esta función se ejecuta una sola vez al encenderse el dispositivo. Inicia a la comunicación Serial a una tasa de 96 000 baudios. Mientras esta comunicación no se establece no continua la ejecución del programa, pues es necesaria para que el Arduino Mega pueda enviarle las órdenes correspondientes. Se establece el pin del ESP32 asociado al *Trigger* del ultrasonido como salida y el del *Echo* como entrada. Por último, se inicializa la OLED y el LoRa.

### 10.1.9. Función Loop

```
183 void loop() {  
184     char buffer[2]; //se crea un buffer de caracteres de 2 posiciones  
185     if (Serial.available() > 0) { //si el serial esta activo  
186         int size = Serial.readBytesUntil('\n', buffer, 2); //lee los caracteres que  
187         //entran por el serial hasta la siguiente linea (/n) y los guarda en el buffer (son 2  
188         //bytes)  
189         if (buffer[0] == 'U') { //si recibe una U ejecuta el proceso  
190             material = int(buffer[1])-48; //convierte el segundo char recibido a int  
191             getReadings(material); //lee el dato del ultrasonidos recibido, lo guarda en la  
192             //variable material recibido  
193             sendReadings(material); //envia los datos  
194             Serial.write('F'); //envia por el serial una F para indicarle al arduino que ha  
195             //finalizado  
196             delay(300);  
197         }  
198     }  
199 }
```

Figura 105: Emisor LoRa. Función Loop

Fuente: Elaboración propia

Esta función, que se ejecutará continuamente en bucle, crea primero un *buffer* de tipo *char* de dos posiciones. Si la comunicación Serial está establecida guarda los caracteres que recibe por el puerto USB y los almacena en el *buffer*.

Si el carácter de la primera posición del *buffer* es una “U”, quiere decir que el Arduino Mega quiere que mida y emita la medición al receptor LoRa. Por tanto, primero obtiene el material que está tratando, que debe de estar almacenado en la segunda posición del *buffer*, pero es de tipo *char*, así que se transforma a tipo *int*, restándole 48. Teniendo el tipo de material se le pasa a la función *getReadings* que, como se explicó anteriormente, mide el nivel y lo almacena en la variable correspondiente. Seguidamente envía los datos mediante LoRa llamando a la función *sendReadings*. Por último, envía el carácter ASCII “F” al Arduino para indicarle que ha finalizado su tarea.

## 10.2. Receptor LoRa<sup>35</sup>

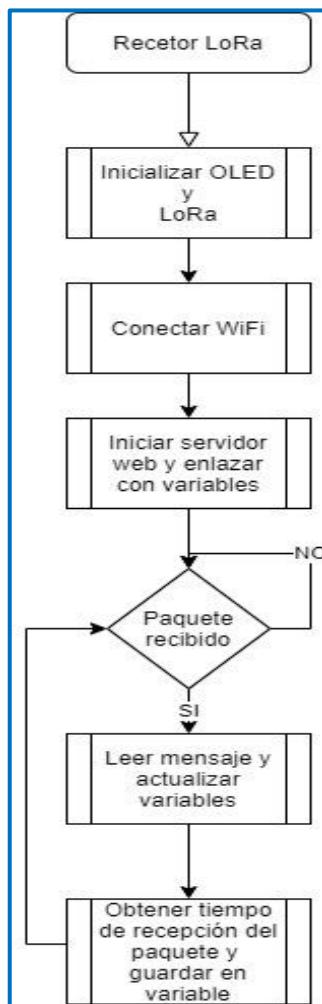


Figura 106: Diagrama de flujo del programa del receptor LoRa

Fuente: elaboración propia

El receptor inicializará primero la pantalla OLED y el LoRa y establecerá después una conexión con el router Wifi para poder generar el servidor web.

Este servidor enlazará unos *placeholder* existentes en el archivo HTML que define el diseño y estructura del servidor (epígrafe 10.3) con variables del programa principal. Las variables son: un **ID** identificativo del paquete recibido; el **tiempo de recepción** del paquete de datos; la calidad de la señal de recepción **RSSI**; y los **5 niveles** de los residuos.

Estas siete variables se actualizarán en el programa principal de la forma que se expone a continuación: si se recibe un paquete, se leerá el mensaje recibido; de él se extraerá el ID del paquete recibido, el nivel y el material al que hace referencia ese nivel; con esto solo se actualizará el nivel de material correspondiente, el resto de los niveles continúan con los valores que tenían previamente (función en el epígrafe 10.2.7); y, al recibir el paquete, también será posible calcular el RSSI. El tiempo de recepción del paquete se obtendrá en una función ejecutada posteriormente (epígrafe 10.2.8).

<sup>35</sup> Código completo del programa en Anexo XX

### 10.2.1. Importación de librerías

```
14 #include <Arduino.h>
15 //Librerias para la conexión WIFI
16 #include <WiFi.h>
17 #include <ESPAsyncWebServer.h>
18
19 #include <SPIFFS.h>
20
21 //Librerias para la gestión del LoRa
22 #include <SPI.h>
23 #include <LoRa.h>
24
25 //Librerias para la visualización el la OLED
26 #include <Wire.h>
27 #include <Adafruit_GFX.h>
28 #include <Adafruit_SSD1306.h>
29
30 // Librerias para el servidor NTP
31 #include <NTPClient.h>
32 #include <WiFiUdp.h>
```

Figura 107: Receptor LoRa. Importación de librerías

Fuente: Elaboración propia

Como el receptor necesitará conectarse a una red WIFI, se incorpora la librería *Wifi.h*. Para la creación del servidor web se necesita la librería *ESPAsyncWebServer.h*. Para almacenar el archivo HTML que contiene el código de diseño de la web, así como de una imagen que se visualizará en dicha web, se necesita recurrir a la librería *SPIFFS.h* (25) que nos permite interactuar con SPIFFS (*SPI Flash File System*), un sistema de archivos diseñado para funcionar en memorias *flash* conectadas por SPI en dispositivos embebidos con escasa cantidad de flash como el ESP32.

Para poder obtener la fecha y hora, datos que se registrarán cada vez que se reciba un paquete, es necesario acceder a un servidor NTP. Por ello se emplea *NTPClient.h*.

La comunicación WIFI se realizará bajo el protocolo UDP (*Universal Datagram Protocol*). Se trata de un protocolo que no requiere respuesta de confirmación de recepción del paquete, lo que lo hace un protocolo menos confiable que el TCP, pero ideal para aumentar la velocidad de transmisión en problemas que requieran transmisión de datos continua o, como es este caso, reducir la carga del servidor y, por tanto, la potencia consumida por el ESP32. Para esta funcionalidad se importa la librería *WiFiUdp.h*.

### 10.2.2. Definición de variables

```
34 //definimos los pines para el modulo LoRa que incorpora el TTGO
35 #define SCK 5 //señal de reloj
36 #define MISO 19 //(Master Input Slave Output) señal de entrada
37 #define MOSI 27 //(Master Output Slave Input) señal de salida
38 #define SS 18 //(Slave Select) selecciona el esclavo al que se envían
39 #define RST 14 //boton de reset
40 #define DIO0 26 //
41
42
43 #define BAND 866E6 //definimos la banda de transmisión en europa es la 866E6
44
45 //Definimos los pines de la OLED
46 #define OLED_SDA 4
47 #define OLED_SCL 15
48 #define OLED_RST 16
49 #define SCREEN_WIDTH 128 // OLED ancho de la pantalla en pixeles
50 #define SCREEN_HEIGHT 64 // OLED alto de la pantalla en pixeles
```

Figura 108: Receptor LoRa. Definición de variables

Fuente: Elaboración propia

Al igual que en el emisor, se declaran la variable de los pines del módulo LoRa y de la OLED, así como la banda de transmisión.

```
52 //definimos los parametros para la conexión WIFI
53 const char* ssid      = "TFG";
54 const char* contrasena = "apruebo";
```

Figura 109: Receptor LoRa. Definición de variables

Fuente: Elaboración propia

Para la conexión WIFI es necesario proporcionar el nombre de la red, así como su contraseña.

```
56 // Definimos el cliente NTP para recibir la fecha y la hora
57 WiFiUDP ntpUDP;
58 NTPClient timeClient(ntpUDP);
```

Figura 110: Receptor LoRa. Definición de variables

Fuente: Elaboración propia

Se define un cliente NTP para obtener la fecha y hora.

```
60 // Variables para guardar el dia y la fecha
61 String formattedDate;
62 String dia;
63 String hora;
64 String timestamp;
```

Figura 111: Receptor LoRa. Definición de variables

Fuente: Elaboración propia

Y se definen las variables “formattedDate” en la que se guardará la fecha en el siguiente formato: AAAA-MM-DDTHH:MM:SSZ. Y variables para guardar el día; la hora; y ambas en *timestamp*.

```
67 // Initialize variables to get and save LoRa data
68 int rssI;
69 String loraMessage; //en esta variable se guarda el mensaje recibido
70
71 //variables de niveles de cada residuo
72 String nivel_carton;
73 String nivel_metal;
74 String nivel_plastico;
75 String nivel_vidrio;
76 String nivel_organico;
77
78 String readingID; //ID recibido
```

**Figura 112:** Receptor LoRa. Definición de variables

**Fuente:** Elaboración propia

Se crea una variable rssI (*Received Signal Strength Indicator*) para guardar la fuerza de la señal recibida.

```
80 // Se crea un servidor en el puerto 80
81 AsyncWebServer server(80);
```

**Figura 113:** Receptor LoRa. Definición de variables

**Fuente:** Elaboración propia

Se crea un servidor (*server*) de la clase AsynWebServer en el puerto 80.

```
83 //se crea un objeto display de tipo Adafruit_SSD1306 con las características de la
     pantalla
84 Adafruit_SSD1306 display(SCREEN_WIDTH, SCREEN_HEIGHT, &Wire, OLED_RST);
```

**Figura 114:** Receptor LoRa. Definición de variables

**Fuente:** Elaboración propia

Se crea un display de tipo Adafruit\_SSD1306 con las características de la pantalla.

### 10.2.3. Función inserta valores en la web

```
86 // Esta funcion inserta variable 'var' de tipo String (con los valores de cada
87 //deposito) en los placeholder del HTML
88 String processor(const String& var){
89     if(var == "CARTON/PAPEL"){ //si el placeholder es CARTON/PAPEL devuelve el nivel
90         //del deposito del carton 'nivel_carton'
91         return nivel_carton;
92     }
93     else if(var == "METAL"){
94         return nivel_metal;
95     }
96     else if(var == "PLASTICO"){
97         return nivel_plastico;
98     }
99     else if(var == "VIDRIO"){
100        return nivel_vidrio;
101    }
102    else if(var == "ORGANICO"){
103        return nivel_organico;
104    }
105
106    else if(var == "TIMESTAMP"){
107        return timestamp;
108    }
109    else if (var == "RSSI"){
110        return String(rssi);
111    }
112    return String();
113 }
```

Figura 115: Receptor LoRa. Función inserta valores en la web

Fuente: Elaboración propia

La función *processor* se encarga de según qué tipo de *placeholder* reciba en la variable “var”, devuelva el nivel de ese residuo, el tiempo o el rssi. De esta forma introduce el valor del nivel, tiempo o rssi en el *placeholder* correspondiente del HTML.

### 10.2.4. Función de inicialización de la OLED

```
//Inicializa la pantalla OLED
void startOLED(){
    //resetea el OLED
    pinMode(OLED_RST, OUTPUT);
    digitalWrite(OLED_RST, LOW);
    delay(20);
    digitalWrite(OLED_RST, HIGH);

    //inicializa el OLED
    Wire.begin(OLED_SDA, OLED_SCL);
    if(!display.begin(SSD1306_SWITCHCAPVCC, 0x3c, false, false)) { // Address 0x3C for
128x32
        Serial.println(F("SSD1306 allocation failed"));
        for(); // Don't proceed, loop forever
    }
    display.clearDisplay(); //limpia la pantalla
    display.setTextColor(WHITE); //establece como blanco el color de escritura
    display.setTextSize(1); //establece el tamaño del texto en una fila
    display.setCursor(0,0); //pone el cursor en el origen (parte superior izquierda)
    display.print("Receptor LoRa");//una vez inicializado escribe en la pantalla este
mensaje
}
```

Figura 116: Receptor LoRa. Función de inicialización de la OLED

Fuente: Elaboración propia

Como en el caso del emisor es necesario inicializar la pantalla, finalmente se muestra el mensaje “Receptor LoRa”.

#### 10.2.5. Función de inicialización del LoRa

```
136 //Inicializa el modulo LoRa
137 void startLoRA(){
138     int counter;
139
140     SPI.begin(SCK, MISO, MOSI, SS); //establece los pines correspondientes
141
142     LoRa.setPins(SS, RST, DIO0); //establece el modulo LoRa
143
144 //hace una prueba para comprobar si se establece conexión en un tiempo determinado
145 while (!LoRa.begin(BAND) && counter < 10) { //envia hasta 10 paquetes mientras no
    haya establecido conexión
146     Serial.print(".");
147     counter++;
148     delay(500);
149 }
150 if (counter == 10) { //si se han enviado los 10 paquetes quiere decir que se agoto
    el tiempo de conexión
151     Serial.println("Error: fallo de inicialización"); //salta un error
152 }
153 Serial.println("Inicializa LoRa OK!"); //si se establecio conexión en la pantalla
    se muestra este mensaje
154 display.setCursor(0,10); //desplaza el cursor 10 columnas a la derecha
155 display.clearDisplay(); //limpia la pantalla
156 display.display(); //lo muestra
157 delay(2000);
158 }
```

Figura 117: Receptor LoRa. Función de inicialización del LoRa

Fuente: Elaboración propia

Al igual que en el emisor, se precisa de una función para inicializar la LoRa.

### 10.2.6. Función de conexión WiFi

```
//funcion para la conexión wifi
void connectWiFi(){

    Serial.print("Conectandose a "); //indica a que WIFI se conecta
    Serial.println(ssid);
    WiFi.begin(ssid, contrasena); //inicia el intento de conexión al wifi con los datos
    proporcionados
    delay(2000);

    while (WiFi.status() != WL_CONNECTED) {//realiza prueba de conexión al wifi
        delay(500);
        Serial.print(".");
    }
    // Print local IP address and start web server
    Serial.println("");
    Serial.println("WiFi conectado.");
    Serial.println("IP address: ");
    Serial.println(WiFi.localIP()); //muestra el IP que se le asignó al ESP32 a través
    del monitor serial
    display.setCursor(0,20); //muestra el cursor a la derecha
    display.print("Accede a la pagina web: ");
    display.setCursor(0,30);
    display.print(WiFi.localIP()); //muestra el IP que se le asignó al ESP32 para que el
    usuario lo introduzca en la APP
    display.display();
}
```

**Figura 118:** Receptor LoRa. Función de conexión WiFi

Fuente: Elaboración propia

Primero intenta conectarse al Wifi con los parámetros de ssid y contraseña establecidos. A continuación, realiza una prueba de conexión. Una vez establecida la conexión, muestra por el monitor Serial y por la OLED la IP que el router asignó al ESP32 y, por tanto, el usuario tendrá que introducir dicha IP en la aplicación para poder visualizar la información.

### 10.2.7. Función de recepción del paquete y extracción de datos

```

184 void getLoRaData() {
185     Serial.print("Paquete LoRa recibido: "); //indica que ha recibido un paquete
186     // lee el paquete
187     while (LoRa.available()) {
188         String LoRaData = LoRa.readString();
189
190         Serial.print(LoRaData);
191
192         //extrae la posición del string LoRaData en el que se encuentra cada uno de los
193         // marcadores que dividen los diferentes datos dentro del mensaje
194         int pos1 = LoRaData.indexOf('/');
195         int pos2 = LoRaData.indexOf('&');
196
197         //extrae los datos del mensaje que estan divididos por los marcadores
198         // anteriormente explicados
199         readingID = LoRaData.substring(0, pos1); //lee desde el origen a la posición 1 y
200         // lo guarda en la variable readingID
201         material = LoRaData.substring(pos1 +1, pos2).toInt(); //recibe el residuo que se
202         // esta clasificando (su posicion en el vector nivel)
203         nivel[material] = LoRaData.substring(pos2+1, LoRaData.length());
204
205     }
206     //muestra el RSSI (received signal strength indicator)
207     rssi = LoRa.packetRssi();
208     Serial.print(" with RSSI ");
209     Serial.println(rssi);
210 }
```

**Figura 119:** Receptor LoRa. Función de recepción del paquete y extracción de datos  
**Fuente:** Elaboración propia

Si se recibe un paquete de datos, se recurre a esta función para extraer el ID del paquete, el nivel y el material al que pertenece este nivel. Para ello es necesario descomponer el mensaje recibido en tres segmentos limitados por los marcadores que se explicaron en el emisor (epígrafe 10.1.6). De forma que, el primer parámetro hasta el marcador ‘/’ corresponde al ID; a partir de este hasta el marcador ‘&’ se encuentra el material que se está procesando y del cual se debe de actualizar su nivel; y, por último, el valor de dicho nivel. Posteriormente se obtiene el valor del RSSI.

### 10.2.8. Función de obtención de la fecha y hora

```

// funcion para obtener la fecha y la hora
void getTimeStamp() {
    while(!timeClient.update()) {
        timeClient.forceUpdate();
    }

    formattedDate = timeClient.getFormattedDate(); //formateamos la fecha y hora
    Serial.println(formattedDate); //lo mostramos

    // Extraemos la fecha
    int splitT = formattedDate.indexOf("T"); //dividimos el string hasta la letra T
    dia = formattedDate.substring(0, splitT); //guardamos el dia
    Serial.println(dia);
    // Extraemos el tiempo
    hora = formattedDate.substring(splitT+1, formattedDate.length()-1);
    Serial.println(hora);
    timestamp = dia + " " + hora;
}
```

**Figura 120:** Receptor LoRa. Función de obtención de la fecha y hora  
**Fuente:** Elaboración propia

Cuando se recibe un paquete se obtendrá la hora y fecha de cuando se recibió. Para ello se emplea la función *getTimeStamp*. Consiste en formatear la fecha y hora obtenida del cliente NTP en el formato AAAA-MM-DDTHH:MM:SSZ. De este formato será entonces sencillo obtener el día dividiendo el *string* hasta la letra “T” y la hora desde la “T” hasta el final. Los dos datos se guardan en la variable *timestamp*.

### 10.2.9. Función Setup

```
void setup() {
    // Initialize Serial Monitor
    Serial.begin(115200); //iniciamos la comunicacion serial a una tasa de 115200
    baudios
    startOLED(); //inicializamos el OLED
    startLoRA(); //inicializamos el LoRa
    connectWiFi(); //conectamos a la red WIFI

    if(!SPIFFS.begin()){ //iniciamos el SPIFFS donde se almacena el html y la imagen
        Serial.println("An Error has occurred while mounting SPIFFS"); //si hay cualquier
        error se indica
        return;
    }

    server.on("/", HTTP_GET, [](AsyncWebRequest *request){//en la pagina
    principal
        request->send(SPIFFS, "/index.html", String(), false, processor); // se muestra
        el index.html y se llama a la función processor para obtener los datos
    });
    server.on("/nivel_carton", HTTP_GET, [](AsyncWebRequest *request){ //en el
    nivel_carton del HTML
        request->send_P(200, "text/plain", nivel_carton.c_str()); //se introducel el
        valor del deposito del carton
    });
    server.on("/nivel_metal", HTTP_GET, [](AsyncWebRequest *request){
        request->send_P(200, "text/plain", nivel_metal.c_str());
    });
    server.on("/nivel_organico", HTTP_GET, [](AsyncWebRequest *request){
        request->send_P(200, "text/plain", nivel_organico.c_str());
    });
    server.on("/nivel_plastico", HTTP_GET, [](AsyncWebRequest *request){
        request->send_P(200, "text/plain", nivel_plastico.c_str());
    });
    server.on("/nivel_vidrio", HTTP_GET, [](AsyncWebRequest *request){
        request->send_P(200, "text/plain", nivel_vidrio.c_str());
    });
    server.on("/timestamp", HTTP_GET, [](AsyncWebRequest *request){
        request->send_P(200, "text/plain", timestamp.c_str());
    });
    server.on("/rss", HTTP_GET, [](AsyncWebRequest *request){
        request->send_P(200, "text/plain", String(rss).c_str());
    });
    server.on("/smartbin", HTTP_GET, [](AsyncWebRequest *request){
        request->send(SPIFFS, "/smartbin.jpg", "image/jpg");
    });

    server.begin(); //inicia el servidor web

    timeClient.begin(); // Inicia NTPClient para obtener el tiempo

    timeClient.setTimeOffset(3600); //para ajustar el tiempo a la zona horaria de
    españa (GMT +1) se aplica un offset de 3600
}
```

Figura 121: Receptor LoRa. Función Setup

Fuente: Elaboración propia

En el *void setup*, que solo se ejecutará una vez al comienzo del programa, se inicializa primero la comunicación serial, la cual es útil a la hora de señalizar errores al programador. Se establece para esta comunicación una tasa de baudios alta de 115 200 baudios. Se ejecutan las funciones de inicialización anteriormente descritas de la OLED y el Lora; se realiza también la conexión Wifi; y se inicializa el SPIFFS notificando cualquier error que ocurriese.

Posteriormente se configura el servidor de forma que cuando reciba diferentes solicitudes HTTP las responda adecuadamente. Es decir, si recibe la solicitud para visualizar la web devuelve el index.html donde encuentra el código que define la web o, por ejemplo, si recibe la solicitud referente al nivel de cartón (“/nivelcarton”), devuelve el nivel de cartón (nivel\_carton.c\_str()).

#### 10.2.10. Función Loop

```
284 void loop() {  
285     //analiza si hay paquetes por recibir  
286     int packetSize = LoRa.parsePacket();  
287     if (packetSize) {  
288         getLoRaData(); //obtiene los datos del mensaje  
289         getTimeStamp(); //obtiene la fecha y la hora  
290     }  
291 }
```

Figura 122: Receptor LoRa. Función Loop

Fuente: Elaboración propia

Esta función se ejecutará continuamente, comprobará si hay paquetes por recibir y, en caso afirmativo, llamará a la función *getLoRaData* para extraer los datos del mensaje e introducirlos en sus variables, así como el RSSI. Lo mismo ocurrirá con la fecha y hora llamando a la función *getTimeStamp*. Una vez obtenidos al haberse enlazado los *placeholder* del HTML con las variables la web se actualizará periódicamente.

### 10.3. Diseño del servidor web<sup>36</sup>

Como se explicó en el receptor se incluye un archivo, index.html, en el cual se describe mediante lenguaje HTML la estructura y diseño del servidor web que el ESP32 alojará.

<sup>36</sup> Código completo del programa en Anexo XX

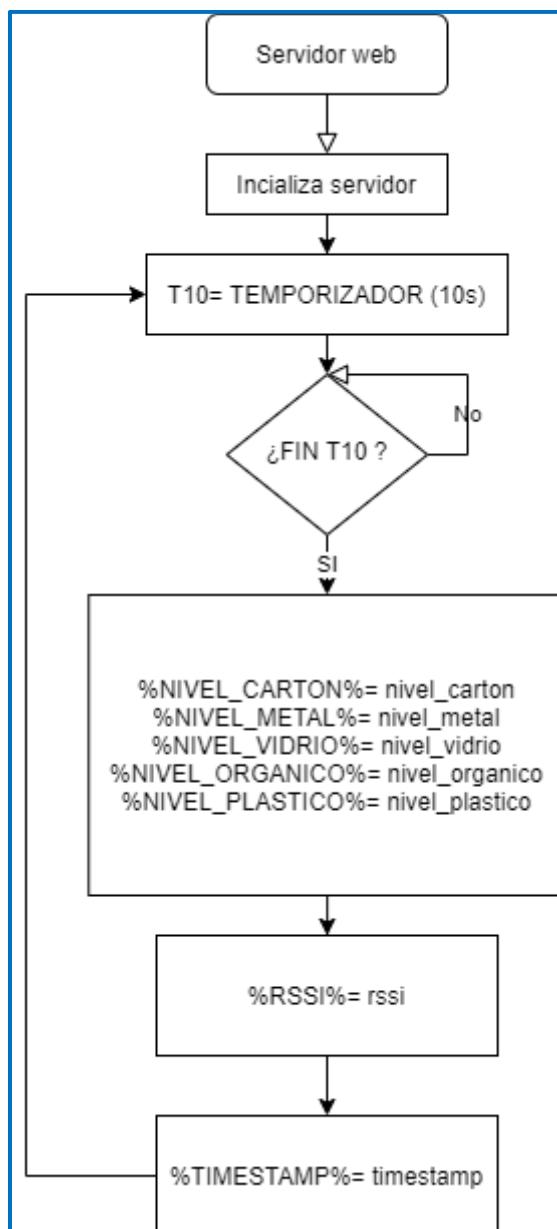


Figura 123: Diagrama de flujo del servidor web

Este estará periódicamente actualizando unos *placeholder* localizados en el HTML con las variables del tiempo de recepción del paquete, el RSSI y los cinco niveles de los residuos.

### 10.3.1. Cabecera

```
1 <!DOCTYPE HTML><html>
2 <head>
3   <meta name="viewport" content="width=device-width, initial-scale=1">
4   <link rel="icon" href="data:,">
5   <title>SMARTBIN DVM</title>
6
7   <script src='https://kit.fontawesome.com/a076d05399.js'></script>
8   <style>
9     body {
10       margin: 0;
11       font-family: Arial, Helvetica, sans-serif;
12       text-align: center;
13     }
14     header {
15       margin: 0;
16       padding-top: 8vh;
17       padding-bottom: 5vh;
18       overflow: hidden;
19       background-image: url(smartbin.jpg);
20       background-size: cover;
21       color: black;
22     }
23     h2 {
24       font-size: 2.0rem;
25     }
26     p { font-size: 1.2rem; }
27     .units { font-size: 1.2rem; }
28     .readings { font-size: 2.0rem; }
29   </style>
30 </head>
```

Figura 124: Servidor web. Cabecera

Fuente: Elaboración propia

En esta parte se importa una librería de símbolos y se establece el tipo de letra, fuente y demás parámetros del título que, en este caso, es “SMARTBIN DVM”, así como los márgenes.

### 10.3.2. Cuerpo

```
<body>
  <header>
    <h2>SMARTBIN DVM</h2>
    <p><strong>Último paquete recibido:<br/><span id="timestamp">%TIMESTAMP%</span></strong></p>
    <p>LoRa RSSI: <span id="rssI">%RSSI%</span></p>
  </header>
  <main>

    <p>
      <i style='font-size:24px' class='far'>#xf15b;</i> Nivel Carton/Papel: <span id="nivel_carton" class="readings">%NIVEL_CARTON%</span>
      <sup>#37;</sup>
    </p>
    <p>
      <i style='font-size:24px' class='fas'>#x0f76;</i> Nivel Metal: <span id="nivel_metal" class="readings">%NIVEL_METAL%</span>
      <sup>#37;</sup>
    </p>
    <p>
      <i style='font-size:24px' class='fas'>#xf0c3;</i> Nivel Vidrio: <span id="nivel_vidrio" class="readings">%NIVEL_VIDRIO%</span>
      <sup>#37;</sup>
    </p>
    <p>
      <i style='font-size:24px' class='fas'>#xf5d1;</i> Nivel Organico: <span id="nivel_organico" class="readings">%NIVEL_ORGANICO%</span>
      <sup>#37;</sup>
    </p>
    <p>
      <i class="fas fa-angle-double-down" style="color:#e8c14d;"></i> Nivel Plastico:
      <span id="nivel_plastico" class="readings">%NIVEL_PLASTICO%</span>
      <sup>#37;</sup>
    </p>
  </main>
  <script>

    setInterval(updateValues, 10000, "nivel_carton");
    setInterval(updateValues, 10000, "nivel_metal");
    setInterval(updateValues, 10000, "nivel_vidrio");
    setInterval(updateValues, 10000, "nivel_organico");
    setInterval(updateValues, 10000, "nivel_plastico");
    setInterval(updateValues, 10000, "rssI");
    setInterval(updateValues, 10000, "timestamp");

    function updateValues(value) {
      var xhttp = new XMLHttpRequest();
      xhttp.onreadystatechange = function() {
        if (this.readyState == 4 && this.status == 200) {
          document.getElementById(value).innerHTML = this.responseText;
        }
      };
      xhttp.open("GET", "/" + value, true);
      xhttp.send();
    }
  </script>
</body>
</html>
```

Figura 125: Servidor web. Cuerpo

Fuente: Elaboración propia

Se muestra la fecha y la hora del último paquete recibido recurriendo a la variable *timestamp* mediante el *placeholder* %TIMESTAMP%. Lo mismo ocurre posteriormente con el RSSI en el *placeholder* %RSSI% y los 5 niveles de los residuos. Estos últimos irán precedidos de un símbolo que define el tipo de residuo al que pertenece, accediendo a la librería de símbolos importada al comienzo del código.

Después, hay una parte del código en JavaScript entre las etiquetas <script> y </script>, en la cual se establece el intervalo de actualización de los valores mostrados en la web, en este caso es de 10 000 ms, llamando a la función *updateValues*. Es decir, cada 10 segundos se actualizarán los *placeholder* de cada residuo, tiempo, RSSI e ID con sus variables correspondientes.

El resultado de este código en HTML proporciona el siguiente diseño:

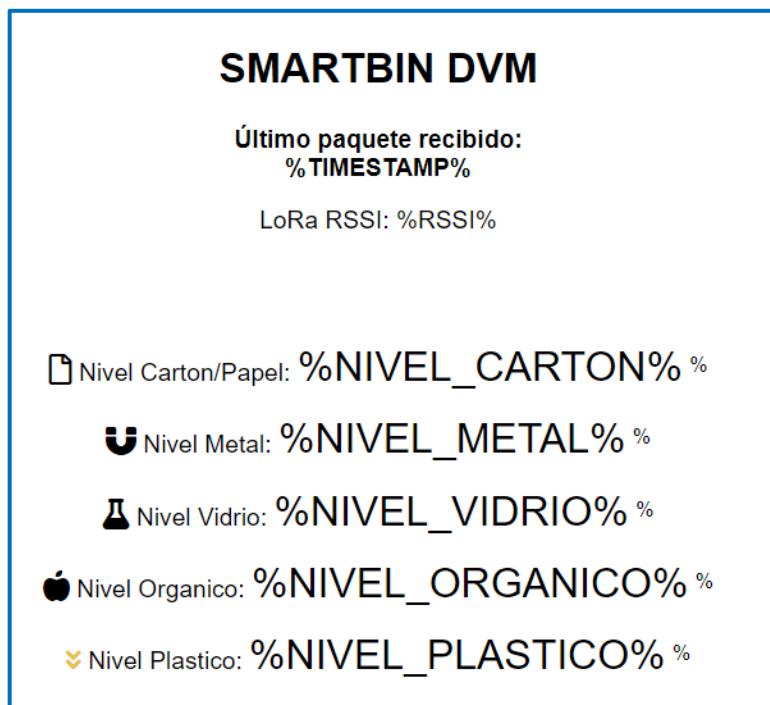


Figura 126: Vista del diseño por HTML

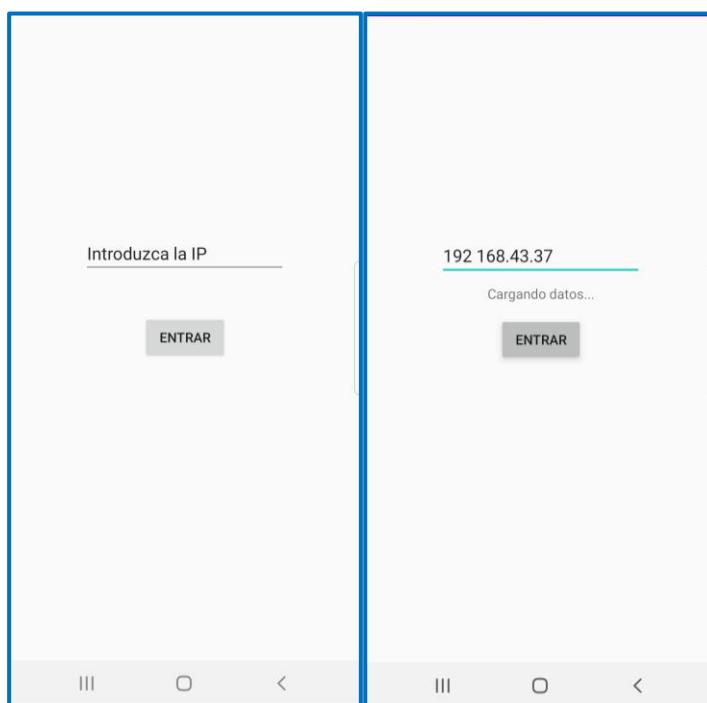
Fuente: Elaboración propia

## 11. DESARROLLO APLICACIÓN MÓVIL

La aplicación móvil no reviste especial complejidad ya que se limitará a acceder al servidor y adaptarlo al formato de la aplicación. No obstante, se decidió incluirla por la relevancia al alza de estos dispositivos móviles y por la capacidad de mejora que posee.

Se desarrolló mediante el lenguaje Java en la plataforma Android Studio. Por tanto, quedará en un principio limitada a dichos dispositivos.

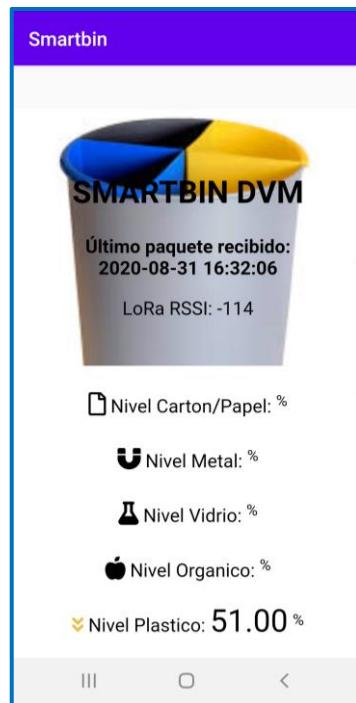
Se compone solamente de dos actividades o pantallas. La primera consiste en una entrada de texto en la que introducir la IP proporcionada por el receptor del LoRa, y un botón “Enviar” para, solamente en el caso de que se introduzca correctamente el formato de IP, poder acceder a la siguiente actividad. Esta acción es indicada mediante un breve texto “Cargando datos...”.



**Figura 127:** Pantalla inicial de la aplicación móvil

**Fuente:** Elaboración propia

En la segunda actividad (*Figura 128*) mediante un “*WebView*” se accede y visualiza el servidor del web alojado en el receptor del LoRa proporcionado en la actividad anterior.



**Figura 128:** Pantalla de visualización de los datos en la aplicación móvil  
**Fuente:** Elaboración propia

## 12. BREVE ESTUDIO ECONÓMICO

A continuación, se expone en la siguiente tabla el coste estimado basándose únicamente en los precios de mercado de cada componente comprado para materializar este proyecto.

**Tabla 3:** Cálculo de costes

**Fuente:** Elaboración propia

| Componente                               | Precio/ud (€) | Uds                | Precio(€) |
|--|---------------|--------------------|-----------|
| Jetson Nano                              | 129           | 1                  | 129       |
| Tarjeta micro sd 32 GB                   | 7             | 1                  | 7         |
| Mega 2560                                | 12,89         | 1                  | 12,89     |
| ESP32 LoRa                               | 11,25         | 2                  | 22,5      |
| LEDs colores (pack 50)                   | 0,035         | 5                  | 0,18      |
| LEDs alta luminosidad (pack 25)          | 0,0656        | 2                  | 0,13      |
| Cables jumper (pack 40)                  | 0,22475       | 39                 | 8,77      |
| Resistencias (lote 600)                  | 0,00821667    | 11                 | 0,09      |
| Condensador 100 µF                       |               | reutilizado        |           |
| Mini BreadBoard                          | 1,05          | 2                  | 2,1       |
| Ultrasonidos HC-SR04                     | 1,8           | 1                  | 1,8       |
| Pi Noir Cam (controlador incl)           | 29,99         | 1                  | 29,99     |
| Motor 28BYJ-48 (controlador incl) pack 5 | 2,398         | 2                  | 4,8       |
| Nema 17                                  | 13,99         | 1                  | 13,99     |
| Controlador A4988 (Nema 17) pack 5       | 1,998         | 1                  | 2         |
| Pulsador rodillo (pack 10)               | 0,564         | 3                  | 1,69      |
| Tubos aluminio 1m                        | 4,99          | 6                  | 29,94     |
| Codo de PVC de Ø160 mm 45º H-M           | 3,39          | 2                  | 6,78      |
| Tablero aglomerado 60x120 cm             | 6,29          | 1                  | 6,29      |
| Bobina PLA reciclada*                    | 18            | 2                  | 36        |
| Spray                                    | 2,87          | 2                  | 5,74      |
| Polea GT2 para motor 16 dientes          | 1,95          | 2                  | 3,9       |
| Rodamiento 6202 2RS                      | 3,63          | 1                  | 3,63      |
| Rodamientos MR83                         | 0,5           | 6                  | 3         |
| Correa GT2 (2m)                          | 2,75          | 1                  | 2,75      |
| Varillas 3 mm (pack 10)                  | 0,965         | 2                  | 1,93      |
| *cálculo aproximado                      |               | <b>Coste total</b> | 336,89    |

Como se puede comprobar en la *Tabla 2*, la realización de este proyecto no requiere de una gran inversión. Y, aunque se trate solamente de un prototipo, su realización a gran escala previsiblemente no incurriría en costes que sobrepasasen en gran medida a las papeleras o contenedores comunes al menos, en lo que a componentes se refiere. De hecho, los componentes más costosos, como se desprende de dicha tabla, serían los electrónicos, pero para desarrollar este proyecto se compraron al por menor, por lo que no se minimizaron los costes. Por ejemplo, en lo que atañe al Jetson Nano, que es el componente cuyo precio es considerablemente más elevado, se podría sustituir adaptando un par de conceptos por una Raspberry PI de apenas 30€. Ya que, siendo conscientes de que la rapidez en la inferencia se vería afectada con esta sustitución, no sería un inconveniente para el caso de esta máquina.

Probablemente los mayores costes sean los de desarrollo y no tanto los de construcción. Pero, de implementarse esta máquina en una red más extensa, posiblemente se amortizaría su precio de una forma más eficiente gracias al ahorro que proporcionaría la optimización de las rutas de recogida, reduciendo costes en combustible, si la recogida se efectuase por una flota de vehículos, y/o mejorando la eficiencia de los trabajadores.

## 13. CONCLUSIÓN Y POSIBLES MEJORAS

Referente a la red neuronal se lograron unos buenos resultados consiguiendo que clasificase los residuos con una precisión de validación máxima del **80,62%**.

En cuanto a la máquina, se presentó el concepto de un dispositivo en el que alojar dicha red neuronal y que estuviese capacitada para realizar los movimientos necesarios para la gestión y clasificación de los residuos. Así como se propuso una estructura organizativa y de comunicación para gestionar de manera fluida los diferentes subprocesos de inferencia sobre la red, ejecución de los movimientos, y comunicación LoRa, distribuidas todas estas tareas en diferentes dispositivos.

Se planteó también una posible y sencilla solución de aplicación móvil para la visualización de los niveles presentes en cada depósito de la máquina.

No obstante, no debe considerarse este proyecto como algo más que un concepto o un planteamiento y, por ende, susceptible de múltiples mejoras como las que se van a enumerar a continuación.

En primer lugar, tal y como se comentó ya en la introducción, cabría la posibilidad de añadir otro tipo de sensores además de la cámara, para alimentar a la red neuronal y tomar decisiones de clasificación más robustas, en base a parámetros de densidad, peso, reflectancia o valor dieléctrico, o incluso lectores de códigos de barras, aunque esto conllevaría una red más compleja y, en el caso de un lector de barras, se necesitaría acceder a una base de datos con las características de cada producto.

Por otro lado, para reducir el coste unitario de cada máquina, se podría plantear la posibilidad de descartar la Jetson Nano y transmitir las imágenes tomadas de los residuos mediante algún tipo de comunicación, por ejemplo, 5G, puesto que, con el LoRa por el momento dado su bajo ancho de banda, resultaría inviable dicha transmisión y, en este caso, no conviene reducir el tamaño de los datos a costa de la calidad de la imagen.

Además, para reducir el volumen de los residuos y facilitar la optimización del espacio en el vehículo de recogida de residuos, se podría implementar un mecanismo de compactación de residuos de bajo consumo, como ya existe en algunos contenedores urbanos. Aunque, si bien es cierto, muchos de los vehículos destinados a la recogida de residuos ya cuentan con su propio sistema de compactación.

Si bien en el interior de la máquina se ha intentado minimizar la relevancia del fondo pintando las paredes y las compuertas del mismo color blanco, se podría aplicar fácilmente un filtro para separar el fondo del residuo en cuestión a la hora de tomar la imagen. Para ello, se tomaría una imagen inicial de referencia y se compararía con la original, obteniendo los píxeles diferentes, es decir la imagen de diferencia. Y con esta imagen “diferencia” se podría alimentar a la red. De todas formas, hay que prestar especial atención a materiales con un color parecido al del fondo, en ese caso habría que ignorar la ejecución de este filtro.

En cuanto a la aplicación móvil, la potenciales mejoras son muchas. Desde añadir un simple botón para apagar la máquina a distancia enviando una variable booleana en un paquete LoRa, hasta una interfaz más adecuada en la cual se podría ofrecer un menú para acceder al servidor del receptor o al servidor con el que cuenta el Jetson Nano.

Pero sin duda la mejora más inmediata y necesaria sería alimentar a la red de un *dataset* más extenso, basado en imágenes tomadas por la propia máquina como se hizo modestamente con el *script* desarrollado para facilitar la creación de ese *dataset* (epígrafe 5.1). Si bien este proyecto se ha beneficiado del poder de generalización del que poseen las redes correctamente entrenadas y, por tanto, también de adaptarse a entornos nuevos en las imágenes, sería muy recomendable la creación de este *dataset*.

## 14. BIBLIOGRAFÍA

- (1) Eurostat. [Ilustración]. 3 de julio de 2020 [consultado 15 julio 2020]. Disponible en: [https://ec.europa.eu/eurostat/databrowser/view/sdg\\_11\\_60/default/table?lang=en](https://ec.europa.eu/eurostat/databrowser/view/sdg_11_60/default/table?lang=en)
- (2) Interempresas. [Ilustración]. 6 de septiembre de 2010 [consultado 15 julio 2020]. Disponible en: <https://www.interempresas.net/Plastico/Articulos/42886-Separacion-de-materiales-ferricos-en-el-reciclaje-de-plasticos.html>
- (3) MEJÍA, C. Llirisaca [et al]. Recolección y clasificación automática de desechos reciclables. En: *Congreso I+D+ingeniería: 2 a 6 de octubre de 2017. Cuenca-Ecuador*. Ingeniería Eléctrica y Electrónica. Maskana, Universidad de Cuenca, vol. 8 (número especial) 2017, pp. 331-340. e-ISSN Nº 2477-8893.
- (4) GOODFELLOW, I., BENGIO, Y. y COURVILLE, A. *Deep Learning*. Cambridge-Massachusetts: Thomas Dietterich (Massachusetts Institute of Technology), 2016. ISBN 9780262035613.
- (5) Dot CSV. ¿Qué es una Red Neuronal? Parte 1: La Neurona/DotCSV [Vídeo]. España: Canal DotCSV; 2018 [consultado 3 junio 2020]. 9 minutos 14 segundos. Disponible en: <https://www.youtube.com/watch?v=MRIV2IwFTPg>
- (6) Machine Learning Mastery. BROWNLEE, J. Loss and Loss Function for Training Deep Learning Neural Networks. 28 enero 2019, revisado octubre 2019 [consultado 15 junio 2020]. Disponible en: <https://machinelearningmastery.com/loss-and-loss-functions-for-training-deep-learning-neural-networks/>
- (7) Raúl Gómez blog [Ilustración]. Plataforma GitHub. 23 mayo 2018 [consultado 12 julio 2020]. Disponible en: [https://gombru.github.io/2018/05/23/cross\\_entropy\\_loss/](https://gombru.github.io/2018/05/23/cross_entropy_loss/)
- (8) Neural Networks: Cross-Entropy [Ilustración]. Plataforma GitHub. ©2017, revisado 2019 [consultado 15 julio 2020]. Disponible en: [https://ml-cheatsheet.readthedocs.io/en/latest/loss\\_functions.html](https://ml-cheatsheet.readthedocs.io/en/latest/loss_functions.html)
- (9) Dot CSV. ¿Qué es el Descenso del Gradiente? Algoritmo de Inteligencia Artificial/DotCSV [Vídeo]. España: Canal DotCSV; 4 febrero 2018 [consultado 18 julio 2020]. 9 minutos 14 segundos. Disponible en: [https://www.youtube.com/watch?v=A6FiCDoz8\\_4](https://www.youtube.com/watch?v=A6FiCDoz8_4)
- (10) HANSEN, C. Optimizers Explained-Adam, Momentum and Stochastic Gradient Descent. *ML From Scratch. Deep Learning* ©2018-2020. 16 octubre 2019 [consultado 15 julio 2020]. Disponible en: <https://mfromscratch.com/optimizers-explained/#/>
- (11) KATHURIA, A. *Intro to optimization in deep learning: Gradient Descent* [Ilustración]. Paperspaceblog. 1 junio 2018 [consultado 30 junio 2020]. Disponible en: <https://blog.paperspace.com/intro-to-optimization-in-deep-learning-gradient-descent/>
- (12) KATHURIA, A. *Intro to optimization in deep learning: Momentum, RMSProp and Adam*. Paperspaceblog. 13 junio 2018 [consultado 7 julio 2020]. Disponible en: <https://blog.paperspace.com/intro-to-optimization-in-deep-learning-momentum-rmsprop-and-adam/>

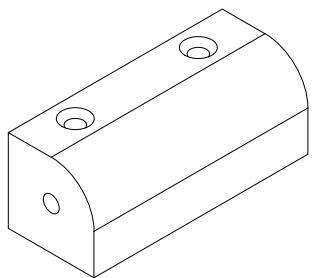
- <https://blog.paperspace.com/intro-to-optimization-momentum-rmsprop-adam/>
- (13) HINTON, G. *Neural Networks for Machine Learning*. Extracto de una clase magistral telemática en la Plataforma Coursera. Disponible en:  
[https://www.cs.toronto.edu/~tijmen/csc321/slides/lecture\\_slides\\_lec6.pdf](https://www.cs.toronto.edu/~tijmen/csc321/slides/lecture_slides_lec6.pdf)
- (14) KINGMA, D. y LEI BA, J. ADAM: a method for stochastic optimization. En: *International Conference on Learning Representations: 7-9 mayo 2015. San Diego*. Ithaka: Cornell University 2015, pp. 1-15. arXiv:1412.6980 [cs.LG].
- (15) DOZAT, T. *Incorporate Nesterov Momentum into Adam*. 2016. Disponible en:  
[http://cs229.stanford.edu/proj2015/054\\_report.pdf](http://cs229.stanford.edu/proj2015/054_report.pdf)
- (16) BIRCANOGLU, C. [et al]. RecycleNet: Intelligent Waste Sorting Using Deep Neural Networks. En: *Computer Science. Innovations in Intelligent Systems and Applications (INISTA)*. Thessaloniki, 2018, pp. 1-7. doi: 10.1109/INISTA.2018.8466276
- (17) THUNG, G. Trashnet. Plataforma GitHub. 10 abril 2017 [consultado 27 julio 2020]. Disponible en:  
<https://github.com/garythung/trashnet>
- (18) CHING, C. *How to build an image classifier for waste sorting*. Towards Data Science. 27 marzo 2019. [Consultado 20 junio 2020]. Disponible en:  
<https://towardsdatascience.com/how-to-build-an-image-classifier-for-waste-sorting-6d11d3c9c478>
- (19) SZEGEDY, C. [et al]. Rethinking the Inception Architecture for Computer Vision. *Computer Science* [en línea]. Diciembre 2015. [consultado 3 agosto 2020]. Disponible en:  
<https://arxiv.org/abs/1512.00567>
- (20) Dataconomy. CULURCIELLO, E. *The history of neural networks*. 19 abril 2017 [consultado 7 agosto 2020]. Disponible en:  
<https://dataconomy.com/2017/04/history-neural-networks/>
- (21) Medium MColombia. TSANG, S. *Review: Inception-v3-1<sup>st</sup> Runner Up (Image Classification) in ILSVRC 2015*. 10 septiembre 2018 [consultado 7 agosto 2020]. Disponible en:  
<https://medium.com/@sh.tsang/review-inception-v3-1st-runner-up-image-classification-in-ilsvrc-2015-17915421f77c>
- (22) Review: Inception-v3-1<sup>st</sup> Runner Up (Image Classification) in ILSVRC 2015 [Ilustración]. MC.AI, septiembre 2018 [consultado 10 agosto 2020]. Disponible en:  
<https://mc.ai/review-inception-v3-1st-runner-up-image-classification-in-ilsvrc-2015/>
- (23) MILTON-BARKER, A. Inception V3 Deep Convolutional Architecture For Classifying Acute Myeloid/Lymphoblastic Leukemia [Ilustración]. Intel: AI Developer Program, 17 febrero 2019 [consultado 3 agosto 2020]. Disponible en:  
<https://software.intel.com/content/www/us/en/develop/articles/inception-v3-deep-convolutional-architecture-for-classifying-acute-myeloidlymphoblastic.html#:~:text=Inception%20V3%20by%20Google%20is,of%20Deep%20Learning%20Convolutional%20Architectures.&text=Inception%20V3%20was%20trained%20for,was%20a%20first%20runner%20up>

- (24) SANTOS, R. *ESP32 LoRa Sensor Monitoring with Web Server (Long Range Communication)*. [consultado 7 agosto 2020]. Disponible en:  
<https://randomnerdtutorials.com/esp32-lora-sensor-web-server/>
- (25) LLAMAS, L. *Como usar el SPIFFS del ESP8266 con el Arduino IDE*. 7 agosto 2019.  
[consultado 8 agosto 2020]. Disponible en:  
<https://www.luisllamas.es/como-usar-el-spiffs-del-esp8266-con-el-arduino-ide/>

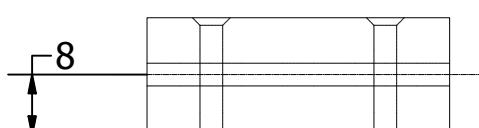
## **15. ANEXOS**

## **15.1. Planos**

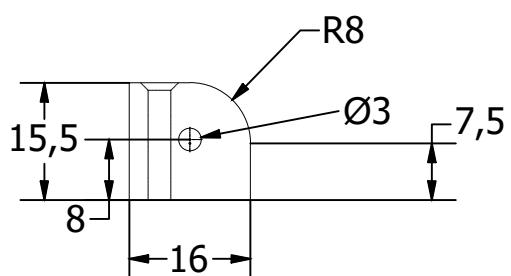
Perspectiva (1:1)



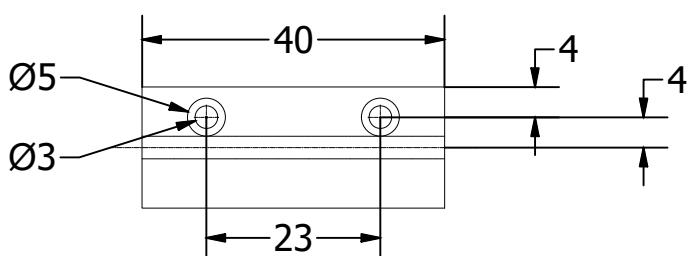
ALZADO



PERFIL IZQ.



PLANTA



|  |  |
|--|--|
| INGENIERO  |  |
| Firmado por VARELA MENENDEZ DANIEL - 45960399M el día 09/09/2020 con un certificado emitido por AC FNMT Usuarios |  |
| Daniel Varela Menéndez   |  |

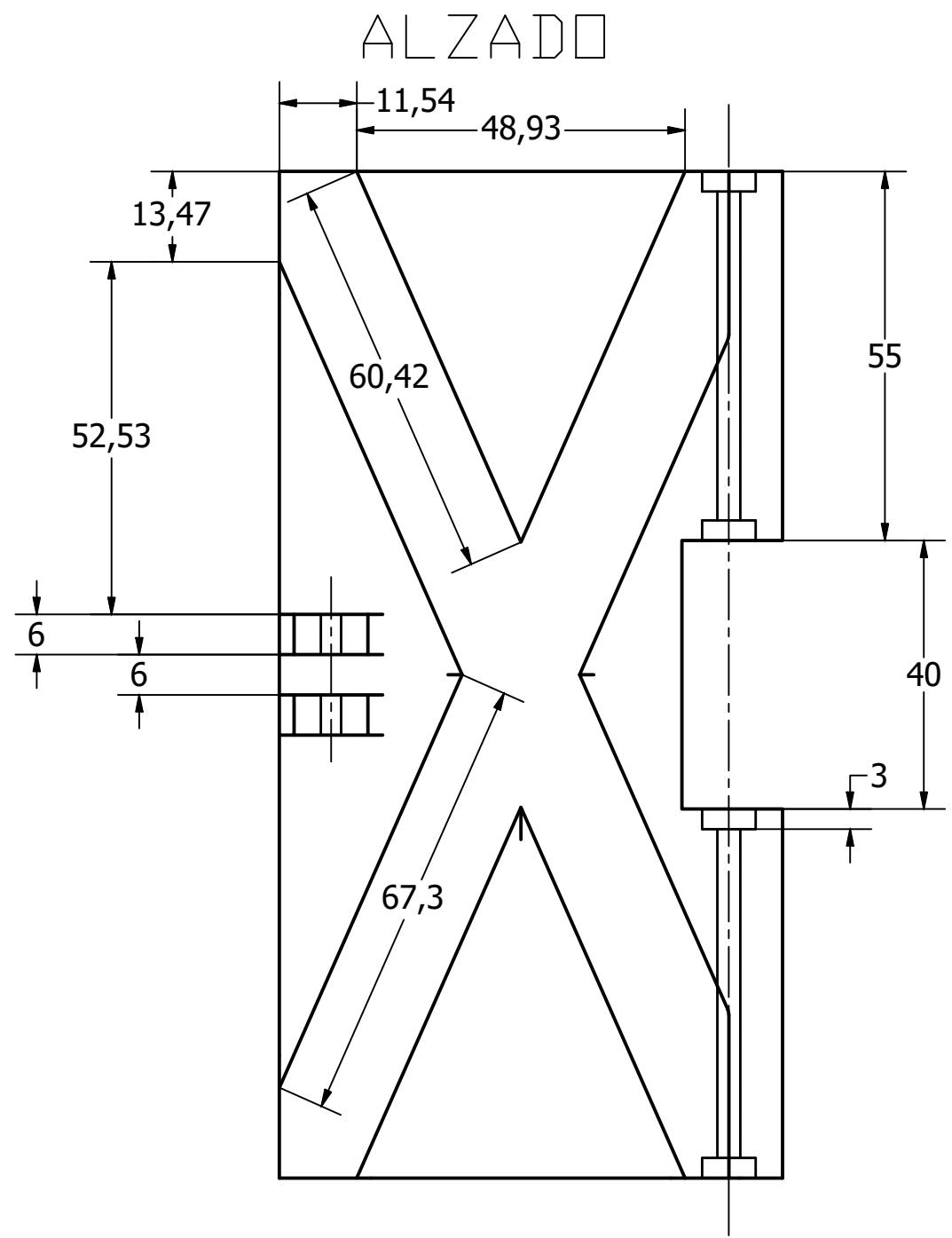
|   |  |
|---|--|
| PROYECTO  |  |
| PRÓYECTO: Clasificación de residuos mediante aprendizaje automático, con gestión centralizada de datos a través de comunicaciones de largo alcance y bajo consumo |  |
| PETICIONARIO: E.E.I. Universidad de Vigo  |  |

|            |          |
|------------|----------|
| REFERENCIA | P1       |
| FECHA      | 19/08/20 |

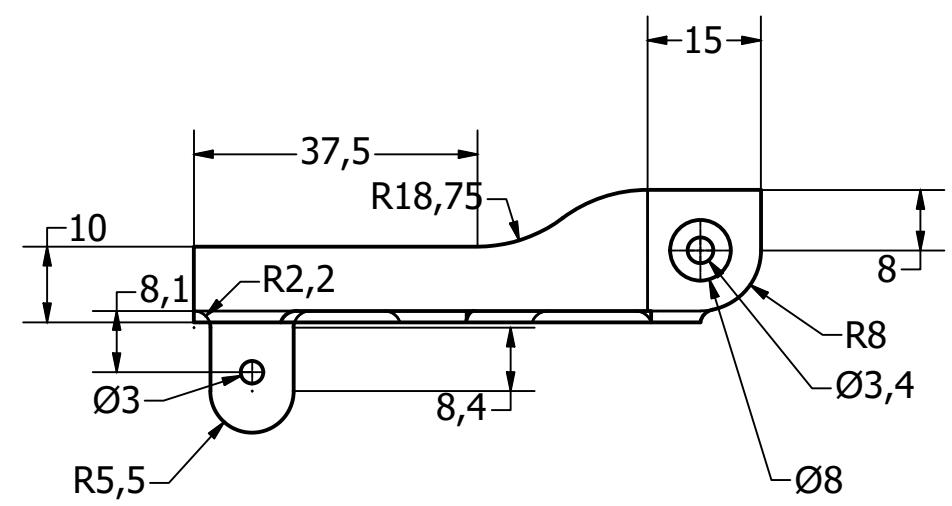
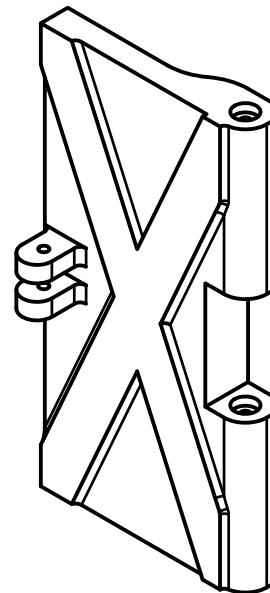
|       |                     |
|-------|---------------------|
| PLANO | Sujección compuerta |
|-------|---------------------|



|             |       |
|-------------|-------|
| ESCALA      | 1:1   |
| Nº DE PLANO | 01/26 |

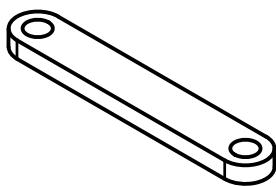


PLANTA

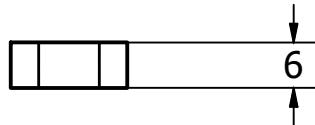
Perspectiva  
(1:2)

| INGENIERO  | PROYECTO  | ESCALA  |
|--|---|---|
| Firmado por VARELA MENÉNDEZ DANIEL - 45960399M el día 09/09/2020 con un certificado emitido por AC FNMT Usuarios | PROYECTO: Clasificación de residuos mediante aprendizaje automático, con gestión centralizada de datos a través de comunicaciones de largo alcance y bajo consumo | <br>Universidade de Vigo |
| Daniel Varela Menéndez   | PETICIONARIO: E.E.I. Universidad de Vigo  | 1:1   |
| REFERENCIA   | PLANO   | Nº DE PLANO   |
| P2   | Compuerta   | 02/26   |
| FECHA  |   |   |
| 19 / 08 / 20   |   |   |

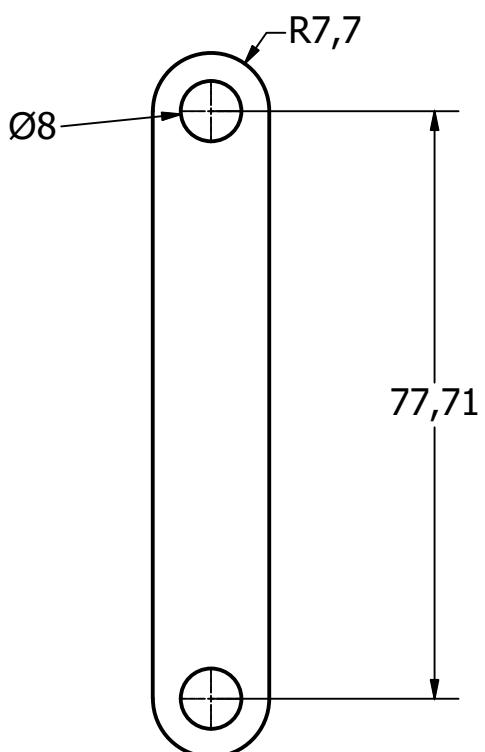
Perspectiva  
(1:2)



## ALZADO



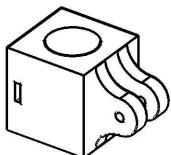
## PLANTA



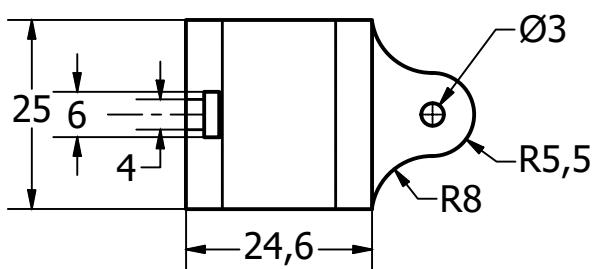
| INGENIERO  | PROYECTO  | PLANO   | ESCALA      |
|--|---|---------|-------------|
| Firmado por VARELA MENENDEZ DANIEL - 45960399M el día 09/09/2020 con un certificado emitido por AC FNMT Usuarios | PROYECTO: Clasificación de residuos mediante aprendizaje automático, con gestión centralizada de datos a través de comunicaciones de largo alcance y bajo consumo<br>PETICIONARIO: E.E.I. Universidad de Vigo | ESLABÓN | 1:1         |
| Daniel Varela Menéndez   |   |         | Nº DE PLANO |
| REFERENCIA   |   |         | 03/26       |
| P3   |   |         |             |
| FECHA  |   |         |             |
| 19/08/20   |   |         |             |



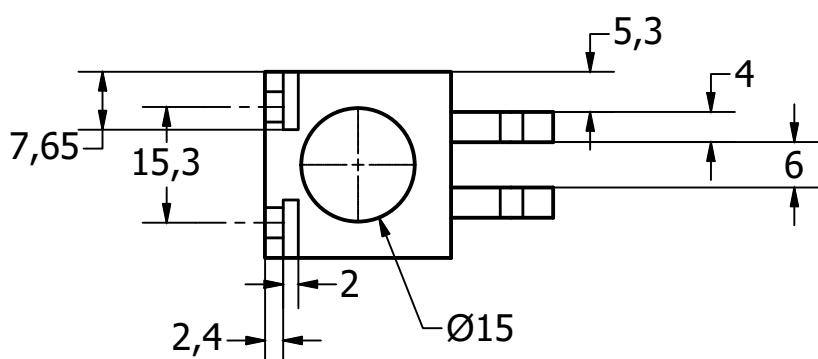
Perspectiva  
(1:2)



## ALZADO



## PLANTA



|  |  |
|--|--|
| INGENIERO  |  |
| Firmado por VARELA MENENDEZ DANIEL - 45960399M el día 09/09/2020 con un certificado emitido por AC FNMT Usuarios |  |
| Daniel Varela Menéndez   |  |

|   |  |
|---|--|
| PROYECTO  |  |
| PROYECTO: Clasificación de residuos mediante aprendizaje automático, con gestión centralizada de datos a través de comunicaciones de largo alcance y bajo consumo |  |
| PETICIONARIO: E.E.I. Universidad de Vigo  |  |

|            |          |
|------------|----------|
| REFERENCIA | P4       |
| FECHA      | 19/08/20 |

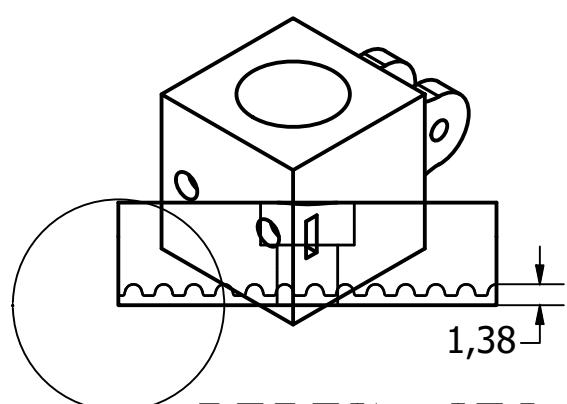
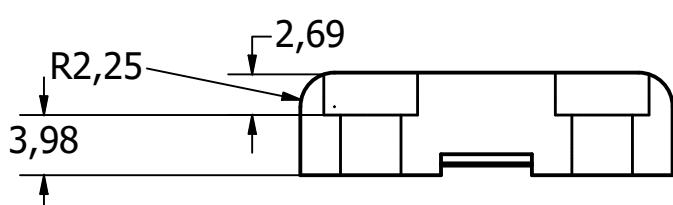
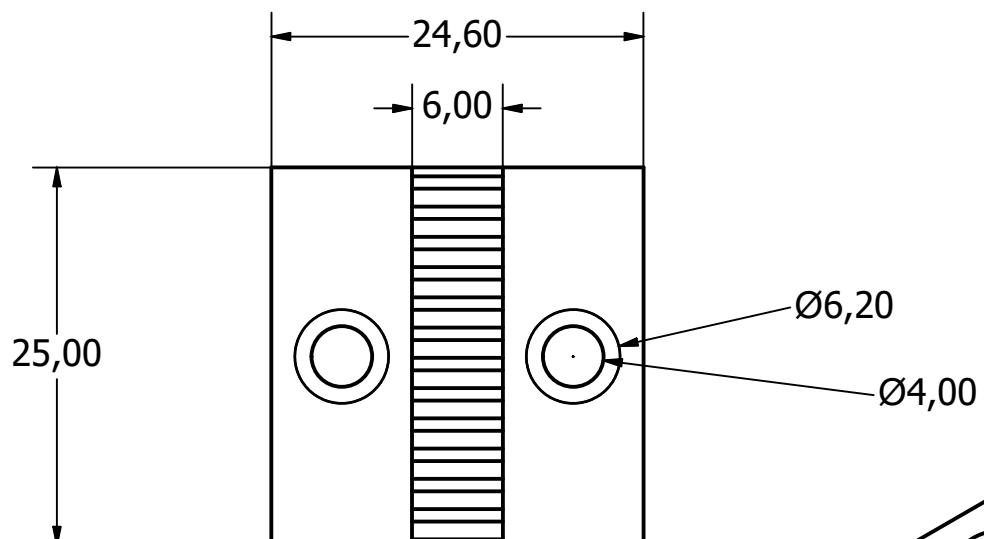
|       |        |
|-------|--------|
| PLANO | Émbolo |
|-------|--------|



|             |       |
|-------------|-------|
| ESCALA      | 1:1   |
| Nº DE PLANO | 04/26 |

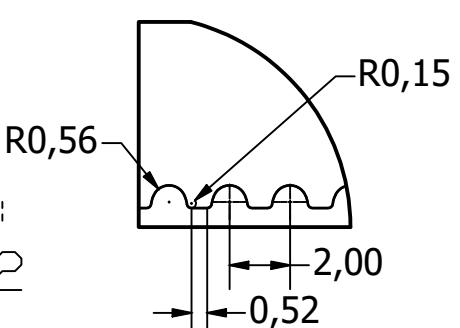
Perspectiva  
(1:1)

## INFERIOR



## ALZADO

## PERFIL IZQ.

DETALLE:  
perfil GT2

## INGENIERO

Firmado por VARELA  
MENENDEZ DANIEL -  
45960399M el día  
09/09/2020 con un  
certificado emitido  
por AC FNMT Usuarios

Daniel Varela Menéndez

## REFERENCIA

P5

## FECHA

19/08/20

## PROYECTO

PROYECTO: Clasificación de residuos  
mediante aprendizaje automático,  
con gestión centralizada de datos  
a través de comunicaciones  
de largo alcance y bajo consumo  
PETICIONARIO: E.E.I. Universidad de Vigo

## PLANO

Sujeción correa



## ESCALA

2:1

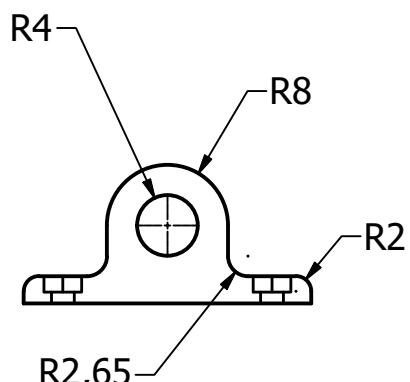
## Nº DE PLANO

05/26

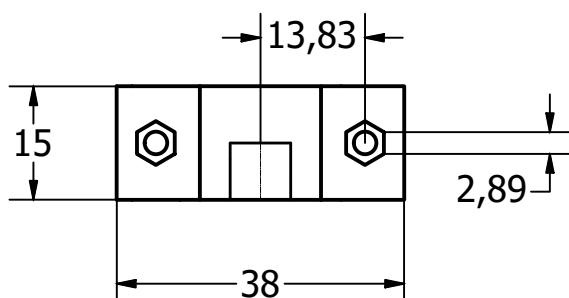
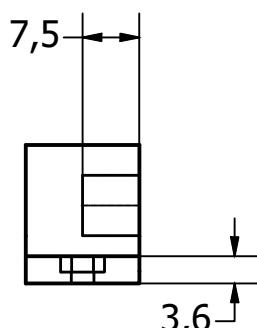
Perspectiva  
(1:2)



ALZADO



PERFIL IZQ



PLANTA

## INGENIERO

Firmado por VARELA  
MENENDEZ DANIEL -  
45960399M el día  
09/09/2020 con un  
certificado emitido  
por AC FNMT Usuarios

Daniel Varela Menéndez

## PROYECTO

PROYECTO: Clasificación de residuos  
mediante aprendizaje automático,  
con gestión centralizada de datos  
a través de comunicaciones  
de largo alcance y bajo consumo  
PETICIONARIO: E.E.I. Universidad de Vigo

## REFERENCIA

P6

## PLANO

Sujeción varillas

## FECHA

19/08/20



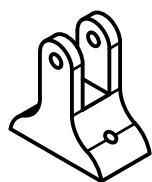
ESCALA

1:1

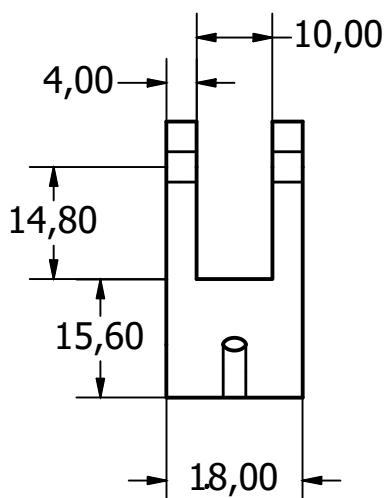
Nº DE PLANO

06/26

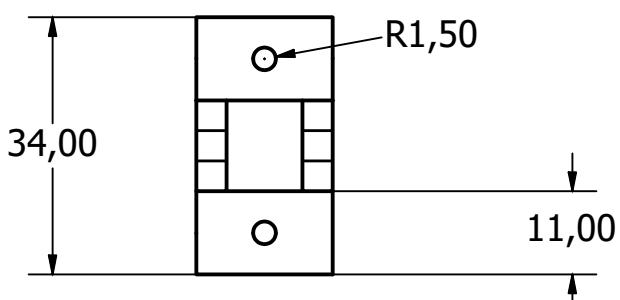
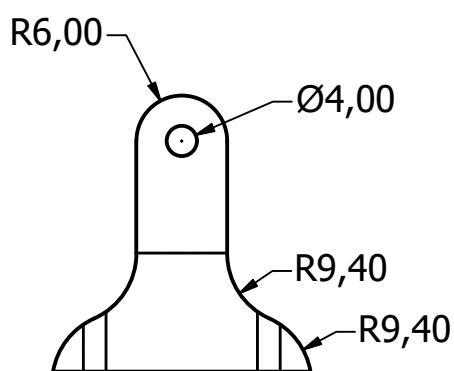
Perspectiva  
(1:2)



ALZADO



PERFIL IZQ



PLANTA

## INGENIERO

Firmado por VARELA  
MENENDEZ DANIEL -  
45960399M el día  
09/09/2020 con un  
certificado emitido  
por AC FNMT Usuarios

Daniel Varela Menéndez

## REFERENCIA

P7

## FECHA

19/08/20

## PROYECTO

PROYECTO: Clasificación de residuos  
mediante aprendizaje automático,  
con gestión centralizada de datos  
a través de comunicaciones  
de largo alcance y bajo consumo  
PETICIONARIO: E.E.I. Universidad de Vigo

## PLANO

Portapoleas

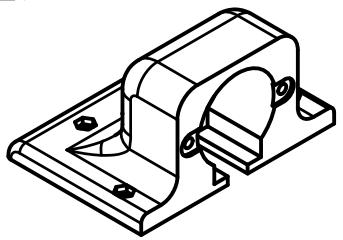


## ESCALA

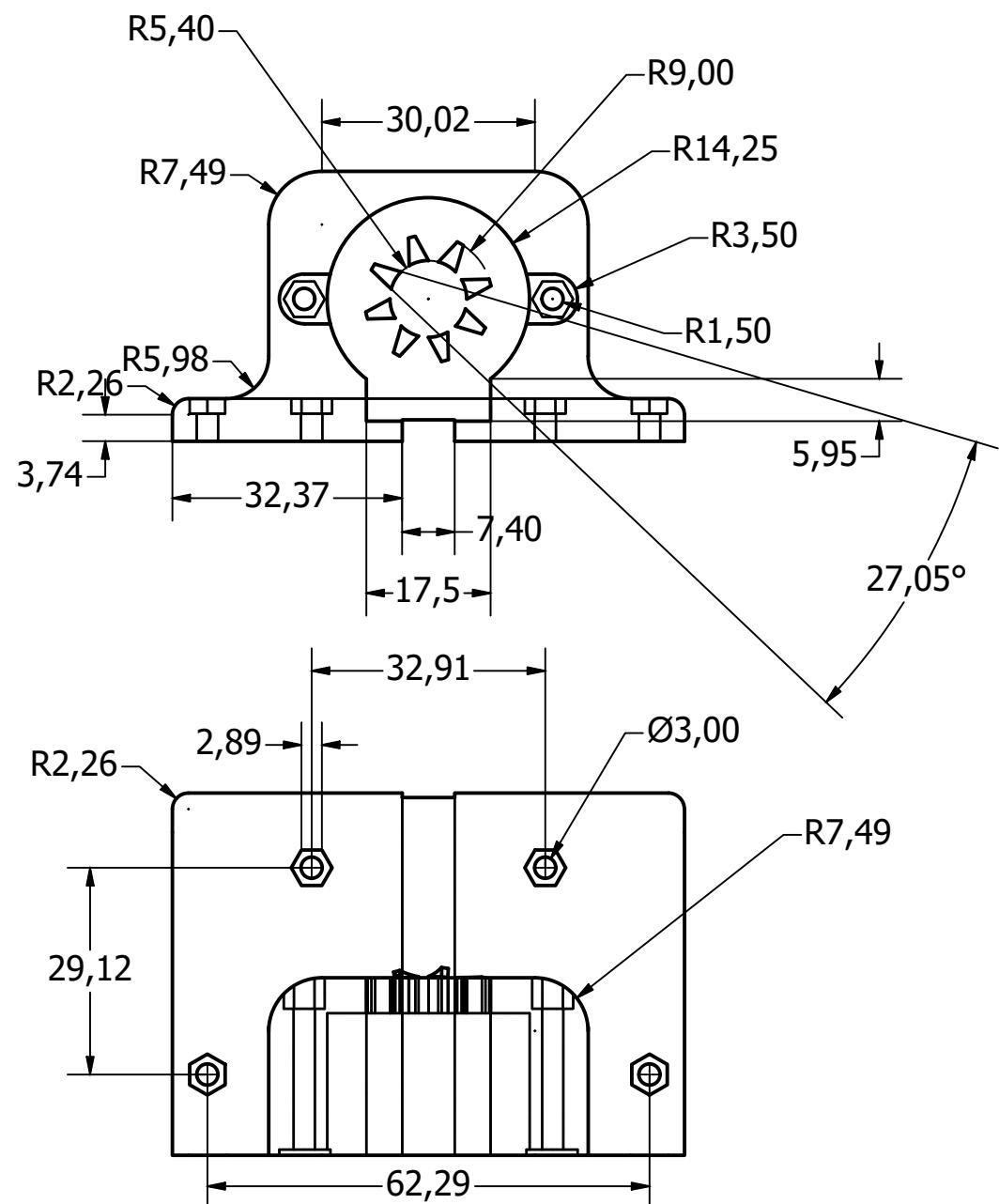
1:1

## Nº DE PLANO

07/26

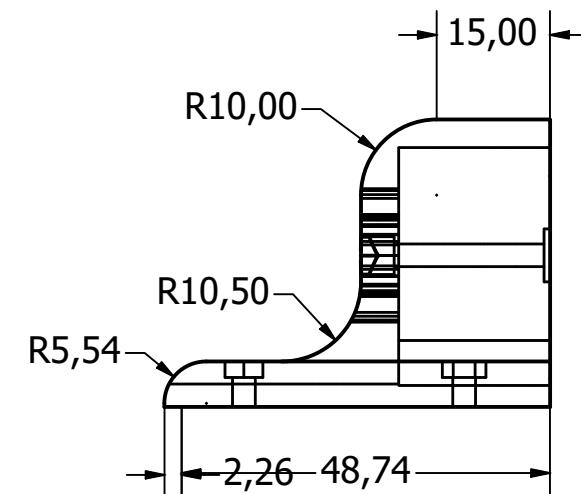
Perspectiva  
(1:2)

## ALZADO



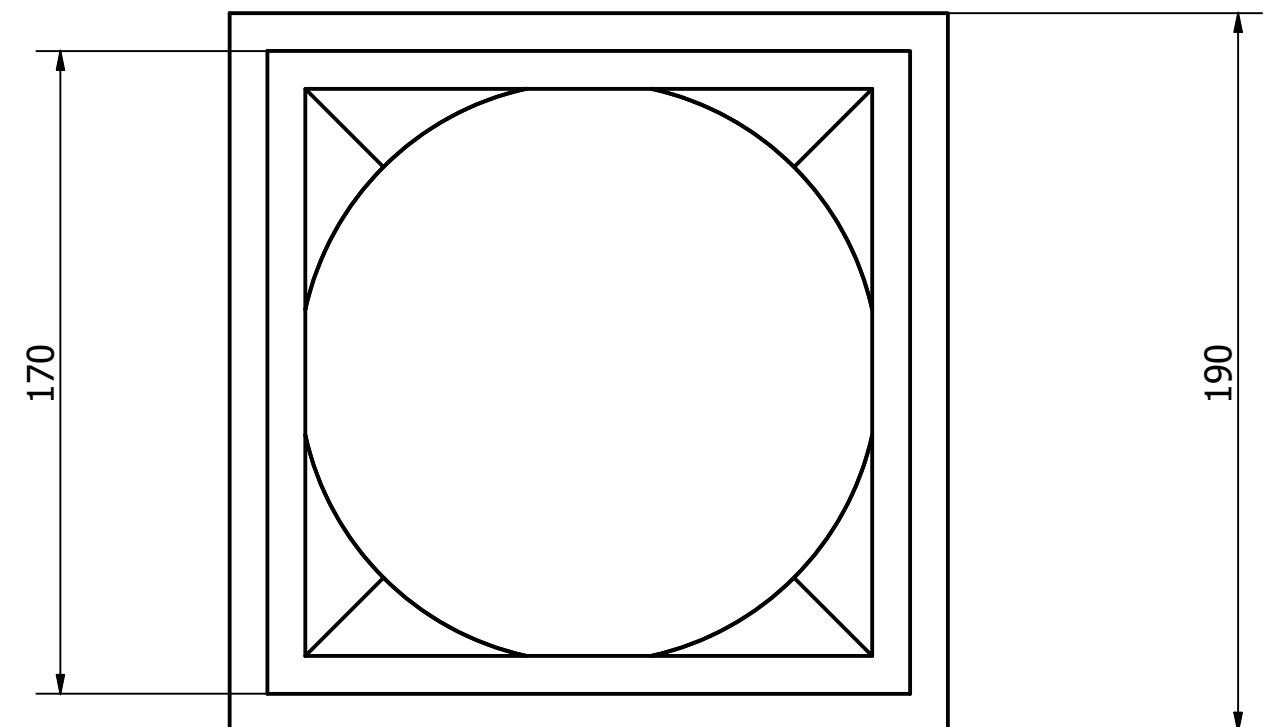
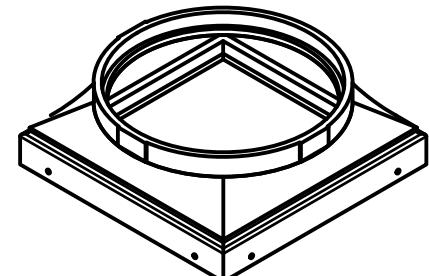
## PLANTA

## PERFIL IZQ.

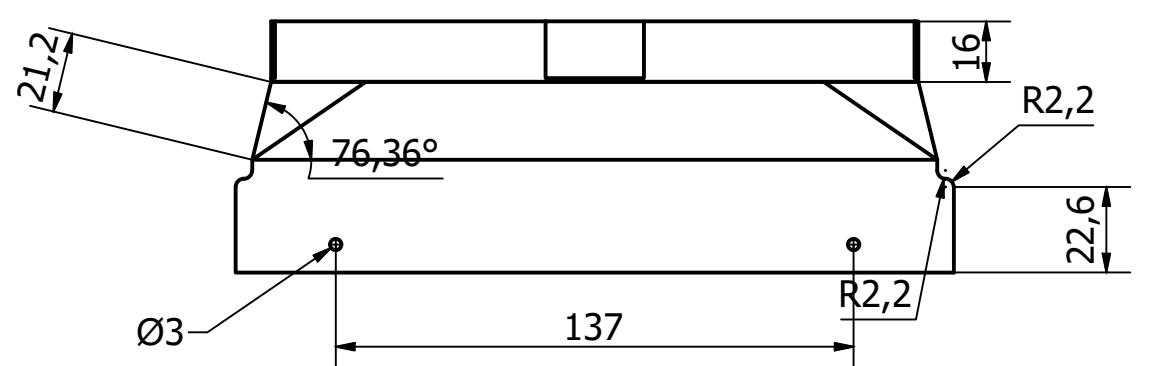


| INGENIERO  | PROYECTO  | ESCALA           |
|--|---|------------------|
| Firmado por VARELA MENENDEZ DANIEL - 45960399M el día 09/09/2020 con un certificado emitido por AC FNMT Usuarios | PROYECTO: Clasificación de residuos mediante aprendizaje automático, con gestión centralizada de datos a través de comunicaciones de largo alcance y bajo consumo | UniversidadeVigo |
| Daniel Varela Menéndez   | PETICIONARIO E.E.I. Universidad de Vigo   | 80               |
| REFERENCIA   | PLANO   | 1:1              |
| P8   | Sujeción motor  | Nº DE PLANO      |
| FECHA  |   | 08/26            |
| 19 / 08 / 20   |   |                  |

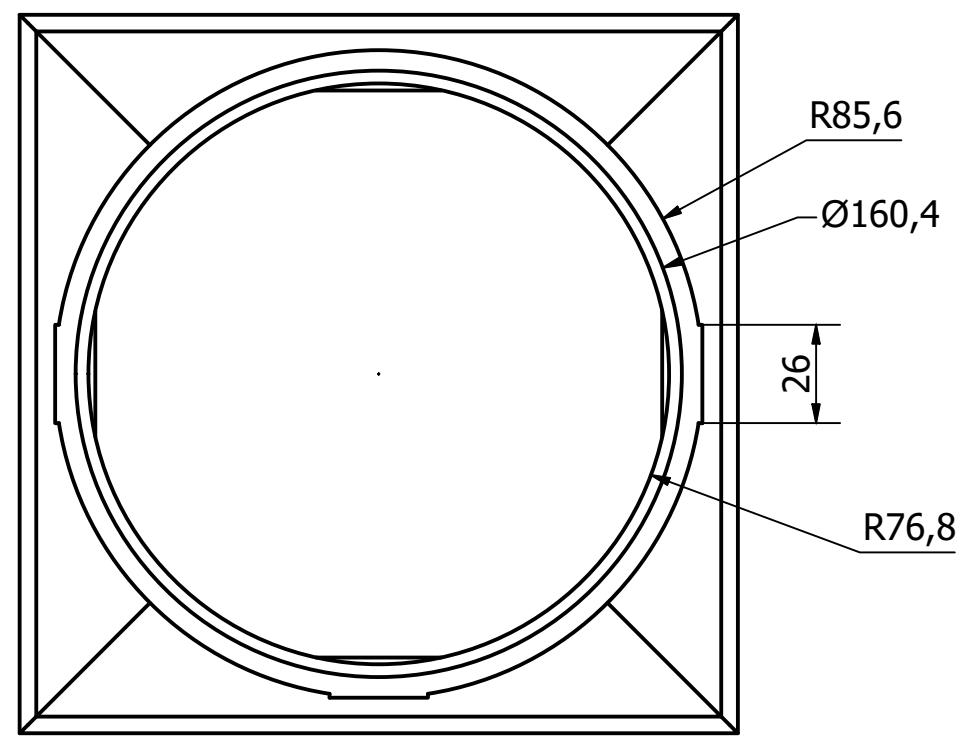
INFERIOR

Perspectiva  
(1:5)

ALZADO



PLANTA

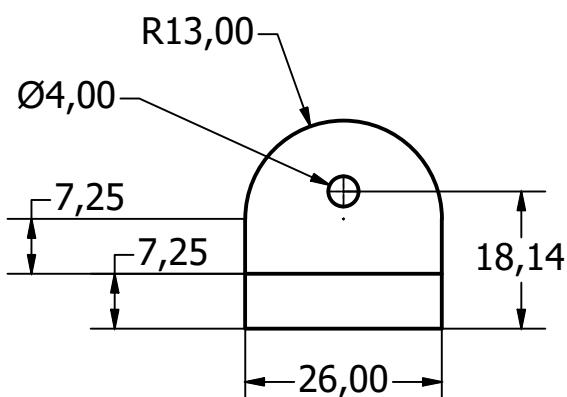


| INGENIERO  | PROYECTO  | ESCALA           |
|--|---|------------------|
| Firmado por VARELA MENENDEZ DANIEL - 45960399M el día 09/09/2020 con un certificado emitido por AC FNMT Usuarios | PROYECTO: Clasificación de residuos mediante aprendizaje automático, con gestión centralizada de datos a través de comunicaciones de largo alcance y bajo consumo | UniversidadeVigo |
| Daniel Varela Menéndez   | PETICIONARIO: E.E.I. Universidad de Vigo  | 1:2              |
| REFERENCIA   | PLANO   | Nº DE PLANO      |
| P9   | Conejero tubo parte sup.  | 09/26            |
| FECHA  |   |                  |
| 19 / 08 / 20   |   |                  |

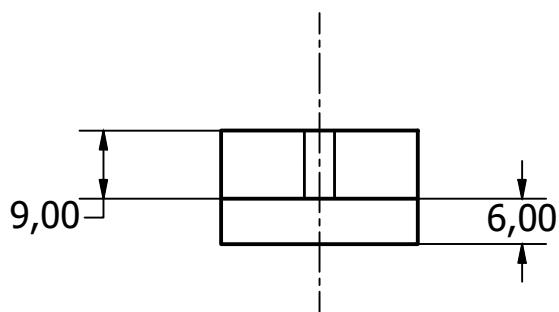
Perpectiva  
(1:2)



## ALZADO □



## PLANTA

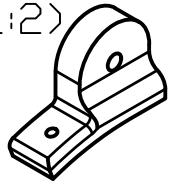


|  |          |
|--|----------|
| INGENIERO  |          |
| Firmado por VARELA MENENDEZ DANIEL - 45960399M el día 09/09/2020 con un certificado emitido por AC FNMT Usuarios |          |
| Daniel Varela Menéndez   |          |
| REFERENCIA   | P10      |
| FECHA  | 19/08/20 |

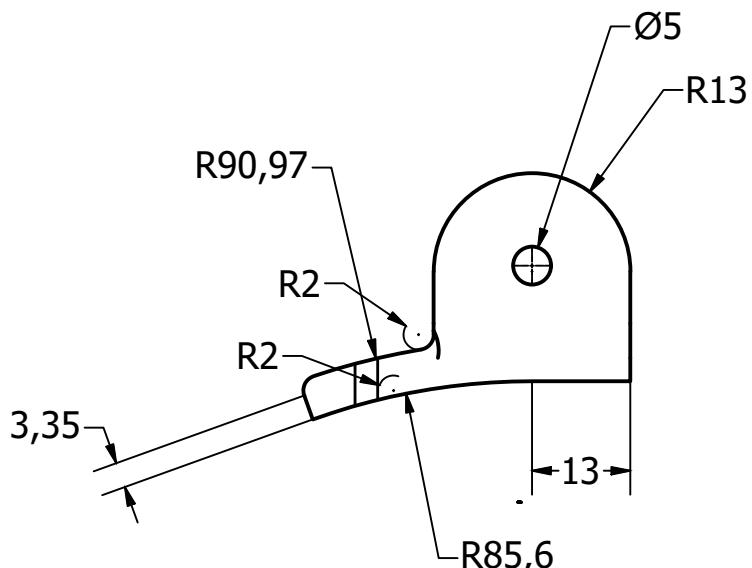
|   |                  |
|---|------------------|
| PROYECTO  |                  |
| PRÓYECTO: Clasificación de residuos mediante aprendizaje automático, con gestión centralizada de datos a través de comunicaciones de largo alcance y bajo consumo |                  |
| PETICIONARIO: E.E.I. Universidad de Vigo  |                  |
| PLANO   | Sujección ruedas |

|             |         |
|-------------|---------|
| ESCALA      | 1:1     |
| Nº DE PLANO | 10 / 26 |

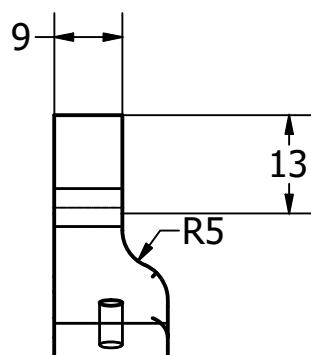


Perspectiva  
(1:2)

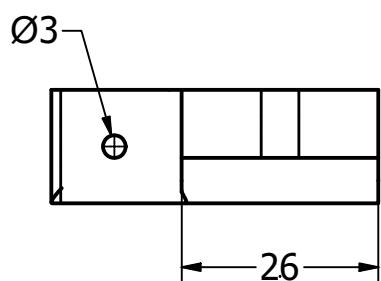
## ALZADO



## PERFIL IZQ.



## PLANTA



## INGENIERO

Firmado por VARELA  
MENENDEZ DANIEL -  
45960399M el día  
09/09/2020 con un  
certificado emitido  
por AC FNMT Usuarios

Daniel Varela Menéndez

## REFERENCIA

P11

## FECHA

19/08/20

## PROYECTO

PROYECTO: Clasificación de residuos  
mediante aprendizaje automático,  
con gestión centralizada de datos  
a través de comunicaciones  
de largo alcance y bajo consumo  
PETICIONARIO: E.E.I. Universidad de Vigo

## PLANO

Porta ruedas  
auxiliar



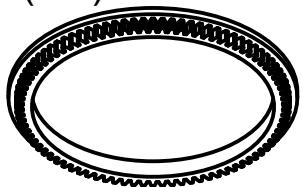
## ESCALA

1:1

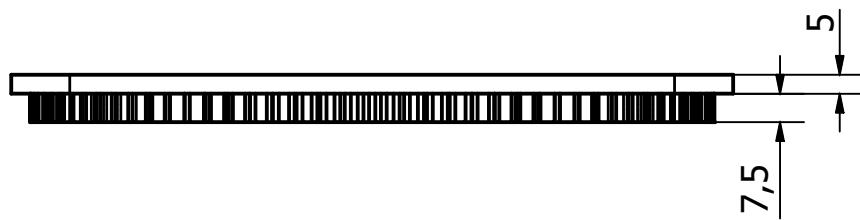
## Nº DE PLANO

11/26

Perpectiva  
(1:5)

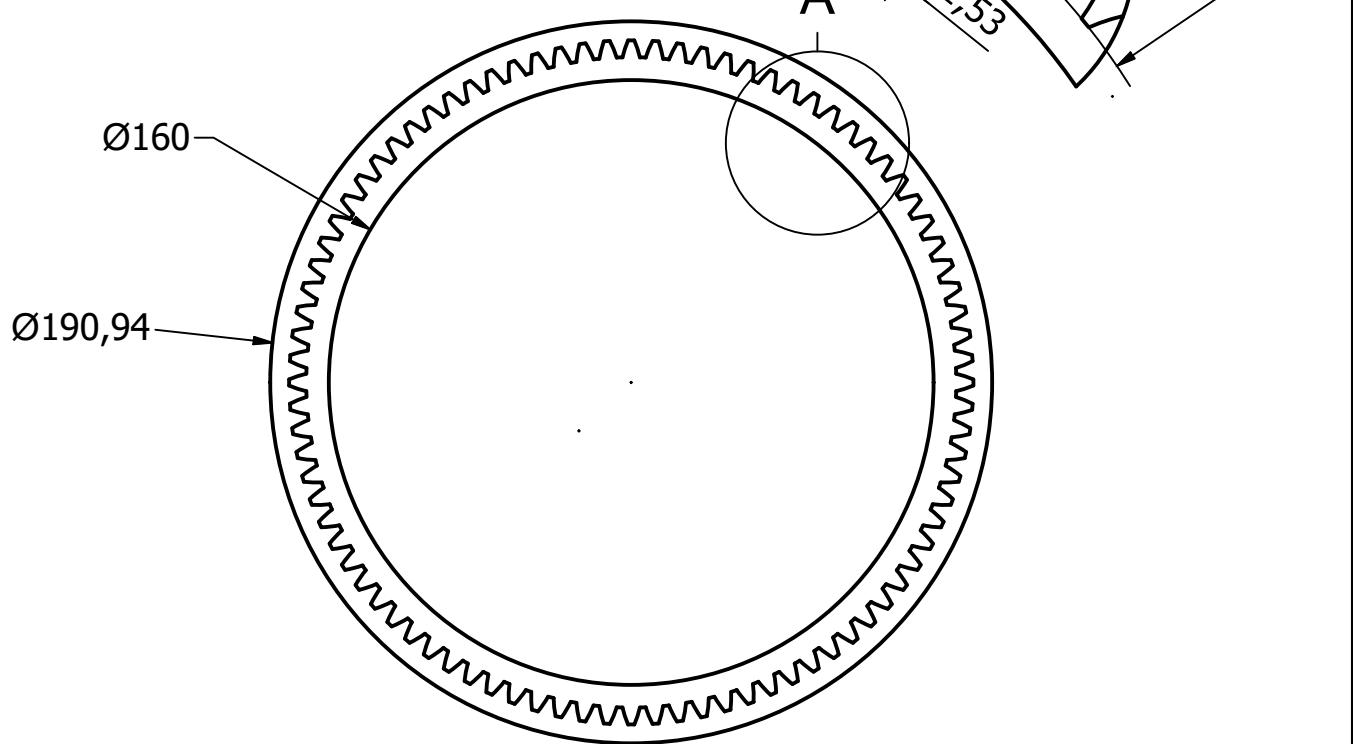


# ALZADO

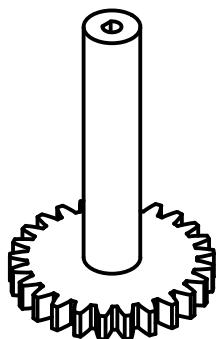


A (1:1)

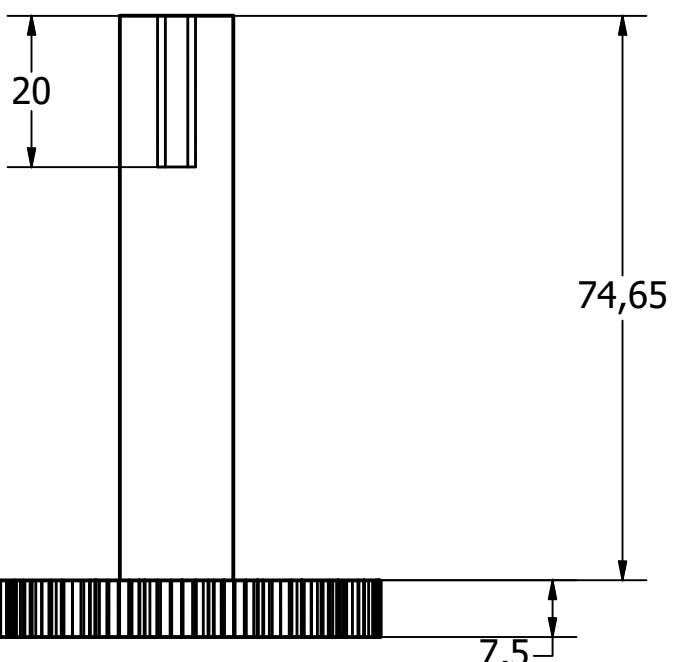
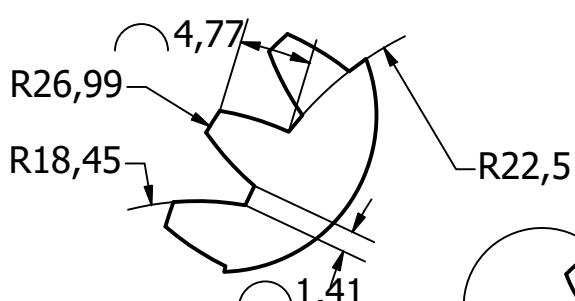
# PLANTA



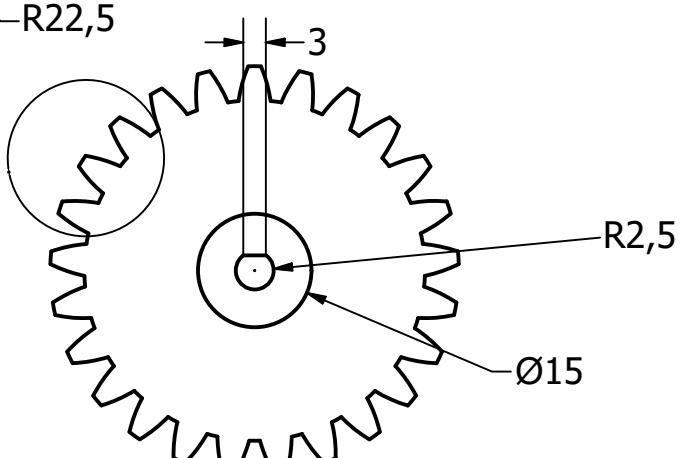
| INGENIERO  | PROYECTO  | UNIVERSIDAD  |
|--|---|--|
| Firmado por VARELA MENENDEZ DANIEL - 45960399M el día 09/09/2020 con un certificado emitido por AC FNMT Usuarios | PROYECTO: Clasificación de residuos mediante aprendizaje automático, con gestión centralizada de datos a través de comunicaciones de largo alcance y bajo consumo<br>PETICIONARIO: E.E.I. Universidad de Vigo | <br>Universidad de Vigo |
| REFERENCIA   | PLANO   | ESCALA   |
| P12  | Engranaje conducido   | 1:2  |
| FECHA  |   | Nº DE PLANO  |
| 19/08/20   |   | 12/26  |

Perspectiva  
(1:2)

## ALZADO

DETALLE A  
ESCALA 2.0000

## PLANTA



## INGENIERO

Firmado por VARELA  
MENENDEZ DANIEL -  
45960399M el dia  
09/09/2020 con un  
certificado emitido  
por AC FNMT Usuarios

Daniel Varela Menéndez

## REFERENCIA

P13

## FECHA

19/08/20

## PROYECTO

PROYECTO: Clasificación de residuos  
mediante aprendizaje automático,  
con gestión centralizada de datos  
a través de comunicaciones  
de largo alcance y bajo consumo  
PETICIONARIO: E.E.I. Universidad de Vigo



## PLANO

Engranaje  
conductor

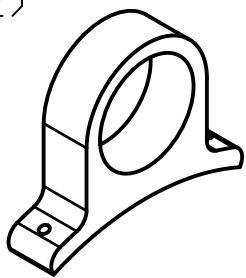
## ESCALA

1:1

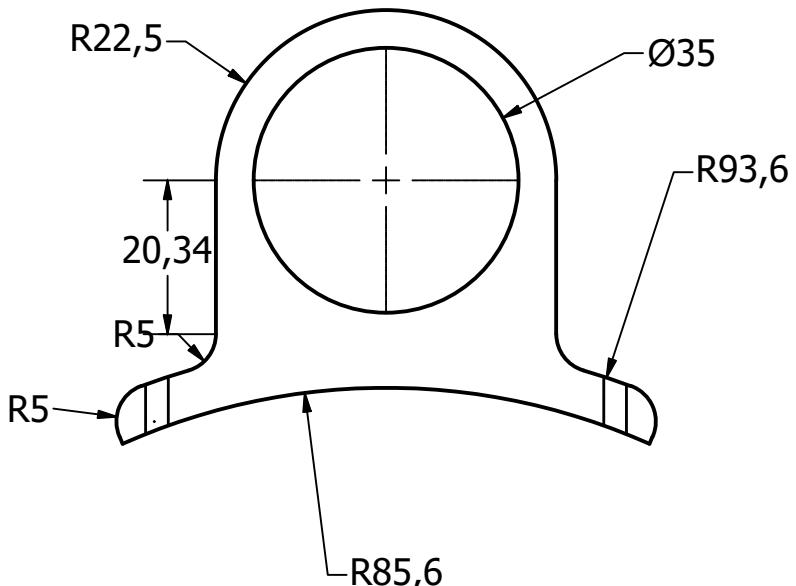
## Nº DE PLANO

13/26

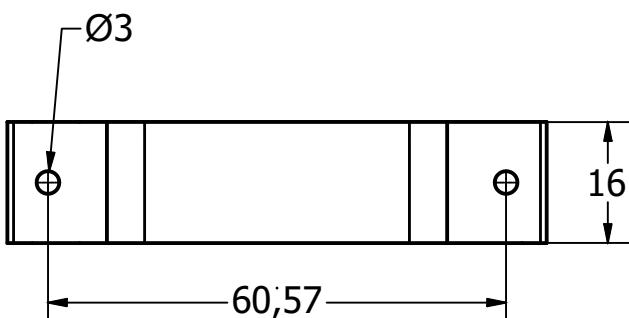
Perpectiva  
(1:2)



## ALZADO



## PLANTA



## INGENIERO

Firmado por VARELA  
MENENDEZ DANIEL -  
45960399M el día  
09/09/2020 con un  
certificado emitido  
por AC FNMT Usuarios

Daniel Varela Menéndez

## REFERENCIA

P14

## FECHA

19/08/20

## PROYECTO

PROYECTO: Clasificación de residuos  
mediante aprendizaje automático,  
con gestión centralizada de datos  
a través de comunicaciones  
de largo alcance y bajo consumo  
PETICIONARIO: E.E.I. Universidad de Vigo

## PLANO

Sujección del  
rodamiento

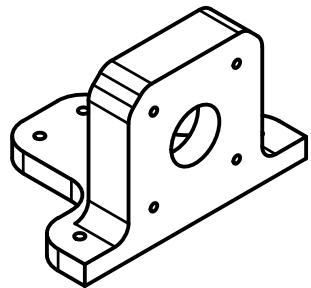


## ESCALA

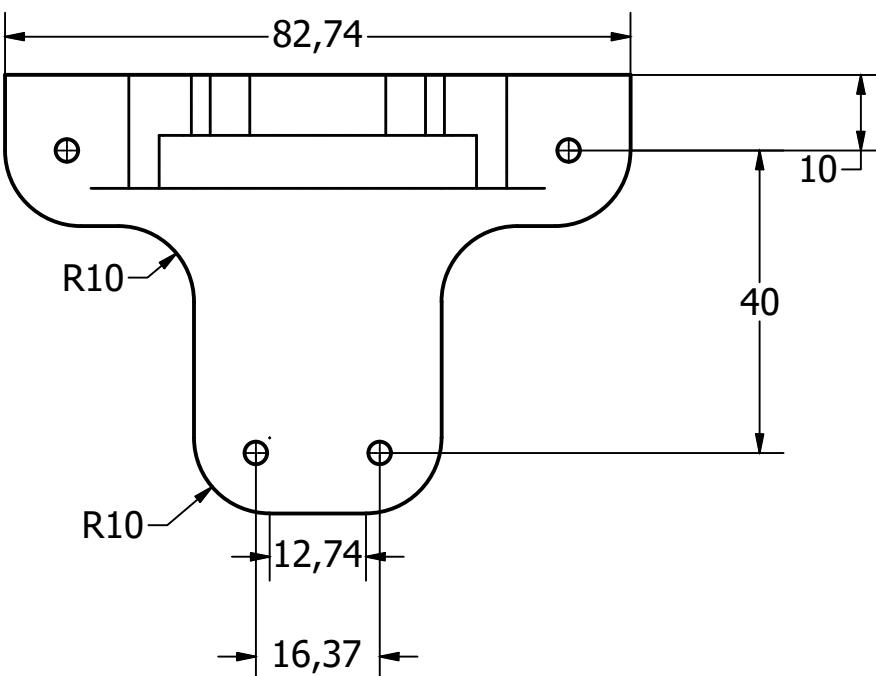
1:1

## Nº DE PLANO

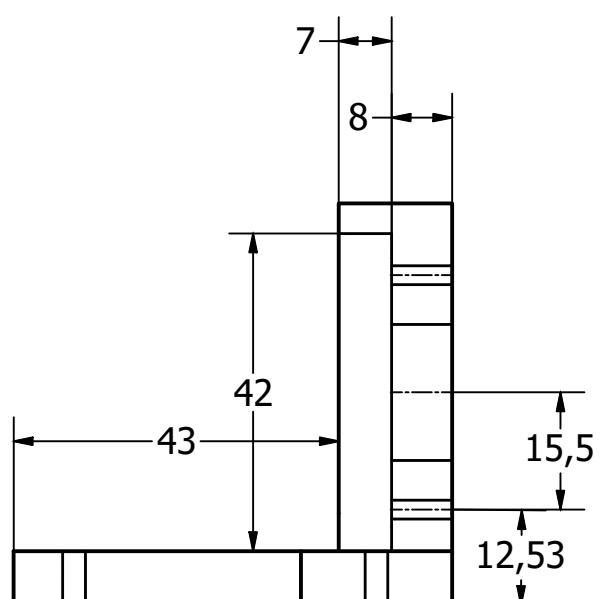
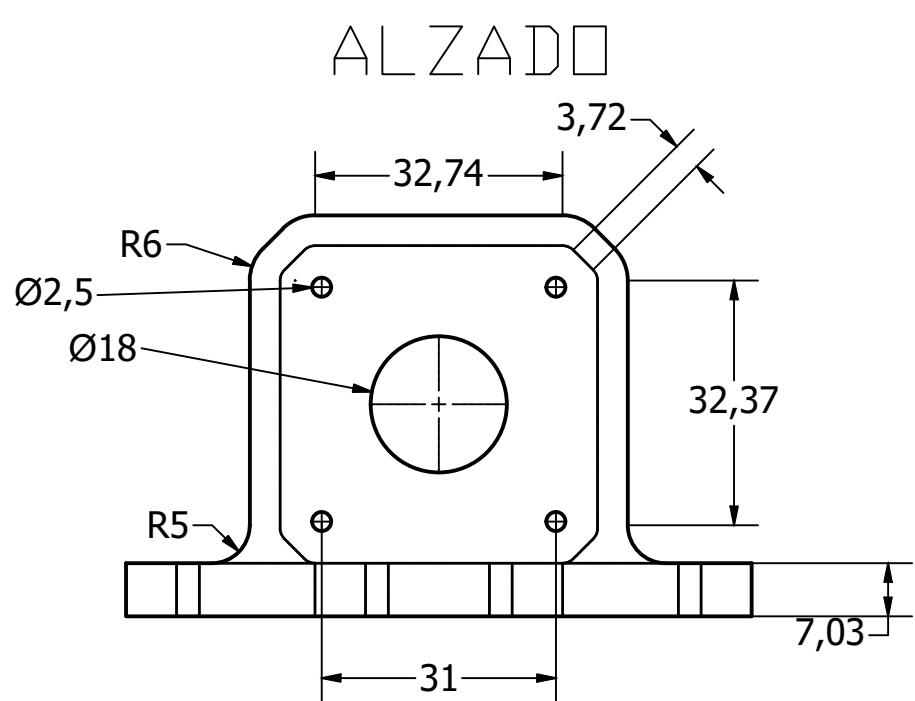
14/26

Perspectiva  
(1:2)

VISTA SUP.

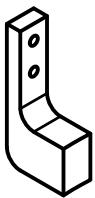


PERFIL IZQ.

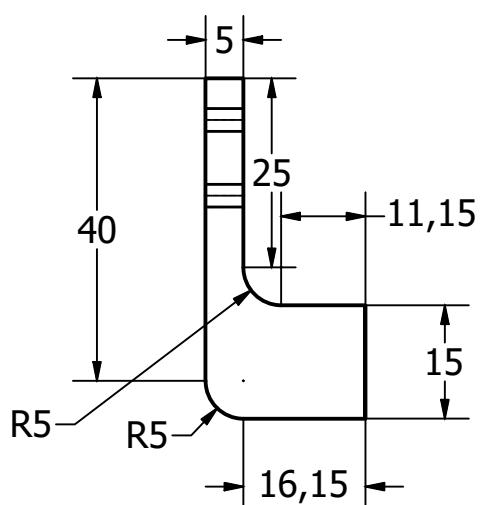


| INGENIERO  | PROYECTO  | ESCALA            |
|--|---|-------------------|
| Firmado por VARELA MENENDEZ DANIEL - 45960399M el día 09/09/2020 con un certificado emitido por AC FNMT Usuarios | PROYECTO: Clasificación de residuos mediante aprendizaje automático, con gestión centralizada de datos a través de comunicaciones de largo alcance y bajo consumo | Universidade Vigo |
| Daniel Varela Menéndez   | PETICIÓNARIOS.E.I. Universidad de Vigo  | 1:1               |
| REFERENCIA   | PLANO   | Nº DE PLANO       |
| P15  | Sujección motor selector  | 15/25             |
| FECHA  |   |                   |
| 19 / 08 / 20   |   |                   |

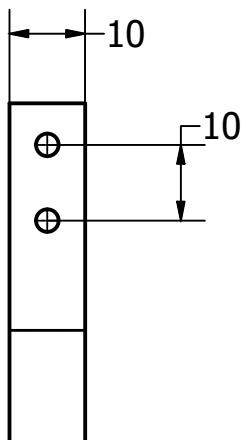
Perspectiva  
(1:2)



ALZADO



PERFIL IZQ.



## INGENIERO

Firmado por VARELA  
MENENDEZ DANIEL -  
45960399M el día  
09/09/2020 con un  
certificado emitido  
por AC FNMT Usuarios

Daniel Varela Menéndez

## REFERENCIA

P16

## FECHA

19/08/20

## PROYECTO

PROYECTO: Clasificación de residuos  
mediante aprendizaje automático,  
con gestión centralizada de datos  
a través de comunicaciones  
de largo alcance y bajo consumo  
PETICIONARIO: E.E.I. Universidad de Vigo

## PLANO

Sujeción Final  
de Carrera



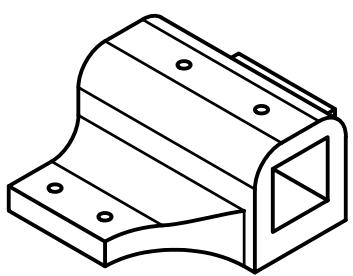
## ESCALA

1:1

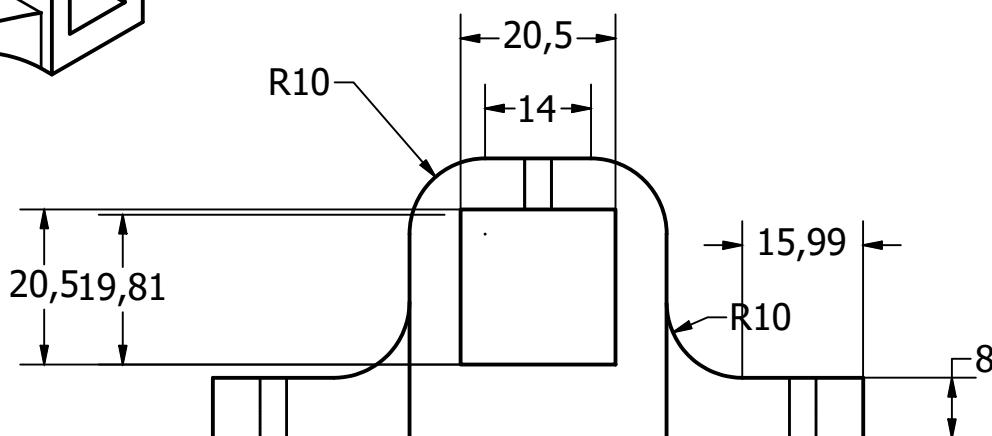
## Nº DE PLANO

16/26

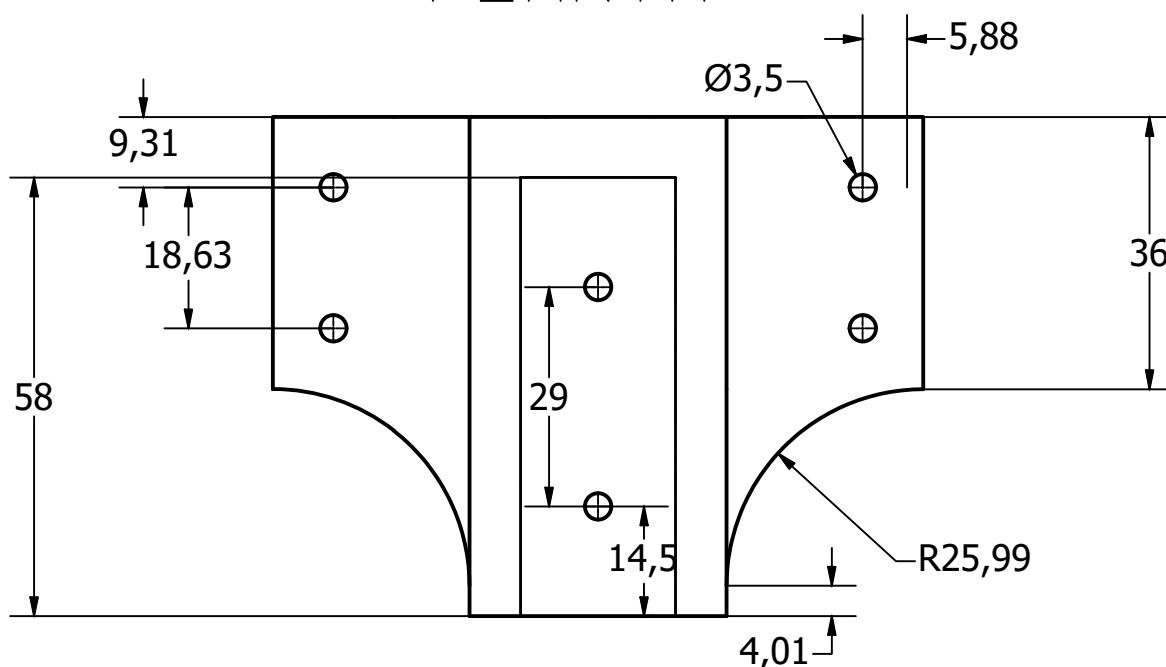
Perspectiva (1:2)



## ALZADO □



## PLANTA



## INGENIERO

Firmado por VARELA  
MENENDEZ DANIEL -  
45960399M el día  
09/09/2020 con un  
certificado emitido  
por AC FNMT Usuarios

Daniel Varela Menéndez

## REFERENCIA

P17

## FECHA

19/08/20

## PROYECTO

PROYECTO: Clasificación de residuos  
mediante aprendizaje automático,  
con gestión centralizada de datos  
a través de comunicaciones  
de largo alcance y bajo consumo  
PETICIONARIO: E.E.I. Universidad de Vigo



## PLANO

SUJECCIÓN TUBO HORIZONTAL

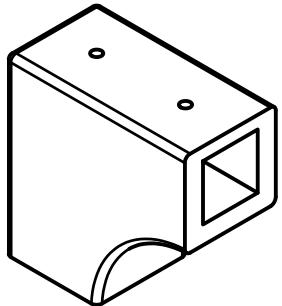
## ESCALA

1:1

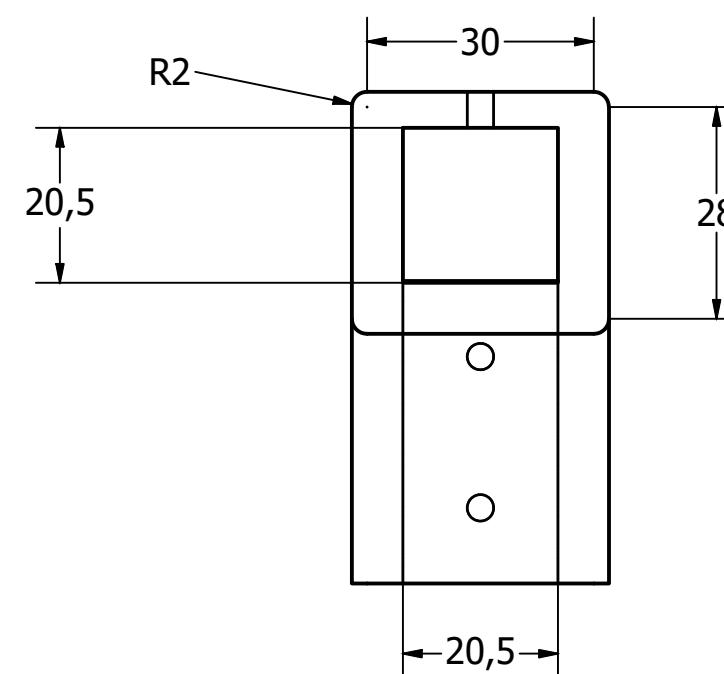
## Nº DE PLANO

17 / 26

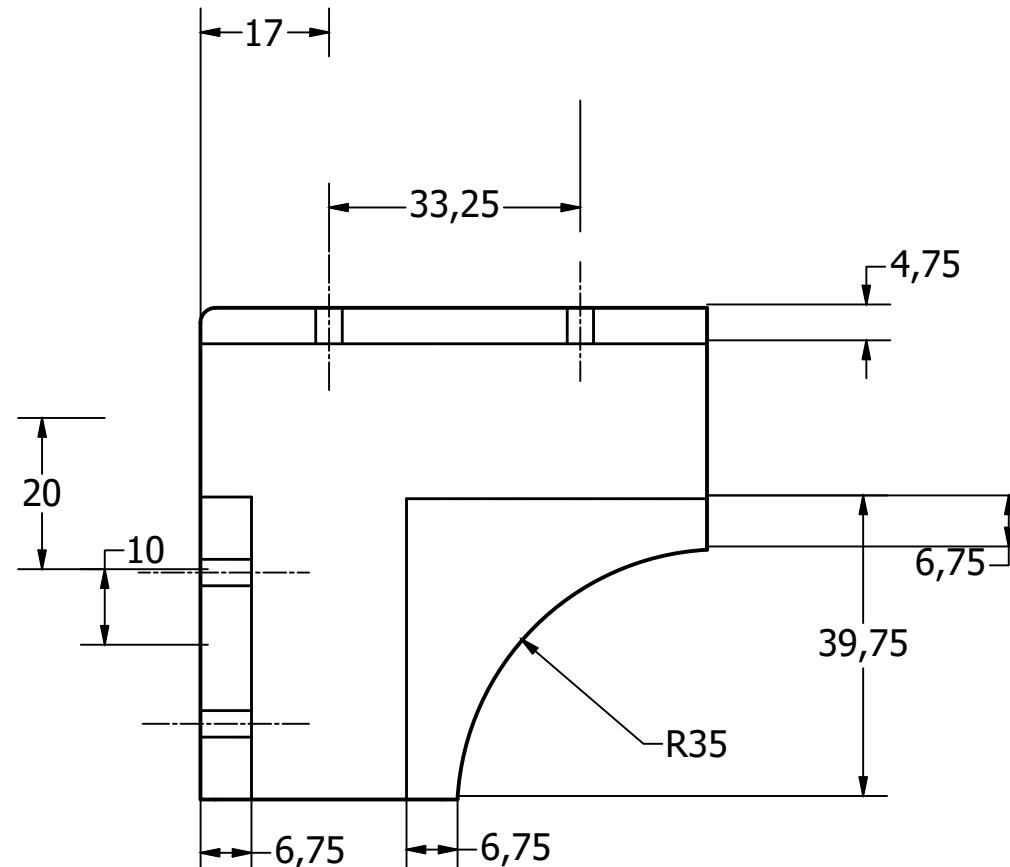
Perspectiva (1:2)



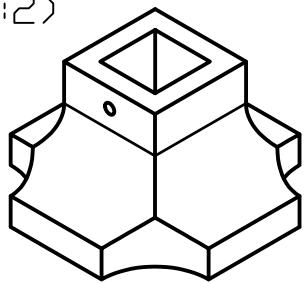
ALZADO



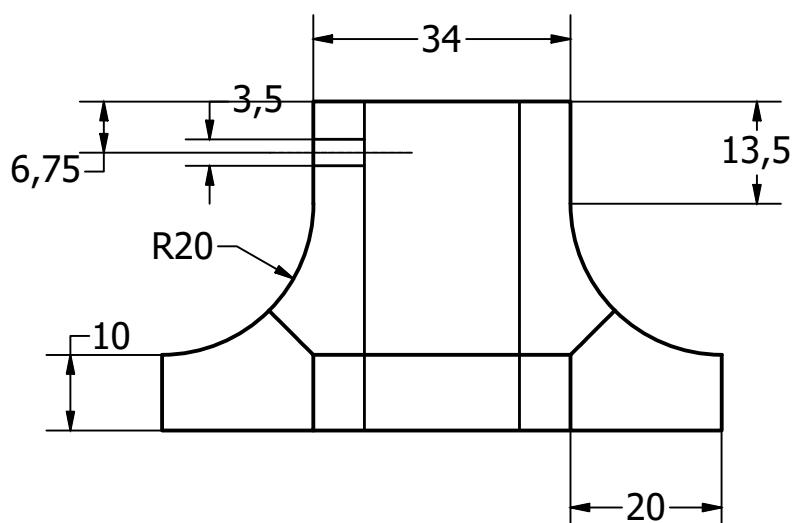
PERFIL IZQ.



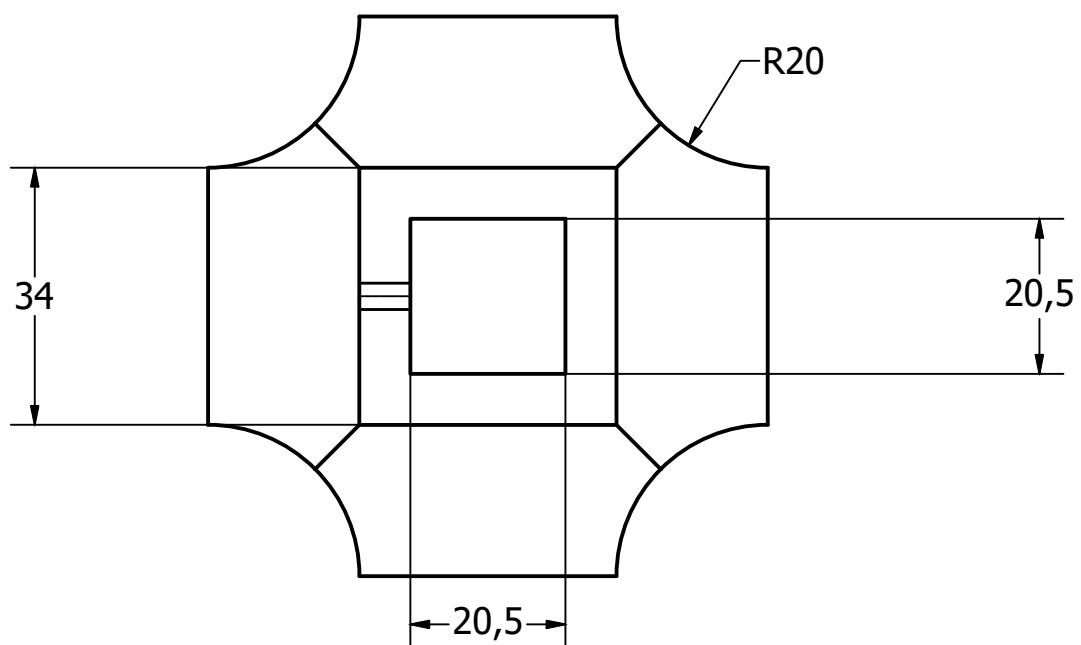
|  |   |             |
|--|---|-------------|
| INGENIERO  | PROYECTO                                |             |
| Firmado por VARELA MENENDEZ DANIEL - 45960399M el día 09/09/2020 con un certificado emitido por AC FNMT Usuarios | PETICIONARIO E.E.I. Universidad de Vigo |             |
| Daniel Varela Menéndez   | PLANO                                   | ESCALA      |
| REFERENCIA   | Codo 90°                                | 1:1         |
| FECHA  | 19 / 08 / 20                            | Nº DE PLANO |
|  |   | 18/26       |

Perspectiva  
(1:2)

## ALZADO



## PLANTA



## INGENIERO

Firmado por VARELA  
MENENDEZ DANIEL -  
45960399M el dia  
09/09/2020 con un  
certificado emitido  
por AC FNMT Usuarios

Daniel Varela Menéndez

## PROYECTO

PROYECTO: Clasificación de residuos  
mediante aprendizaje automático,  
con gestión centralizada de datos  
a través de comunicaciones  
de largo alcance y bajo consumo  
PETICIONARIO: E.E.I. Universidad de Vigo

## REFERENCIA

P19

## FECHA

19/08/20

## PLANO

Pie



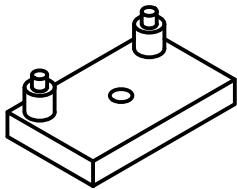
## ESCALA

1:1

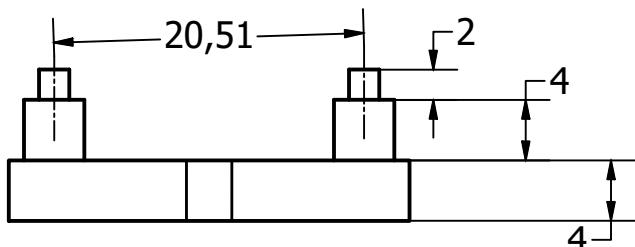
## Nº DE PLANO

19/26

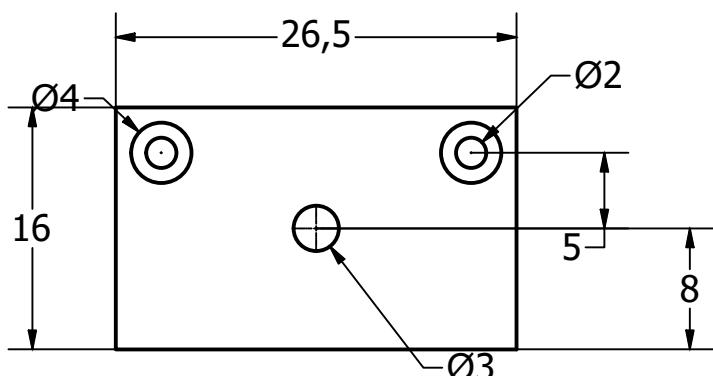
Perspectiva  
(1:1)



## ALZADO

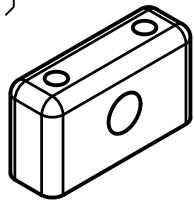


## PLANTA



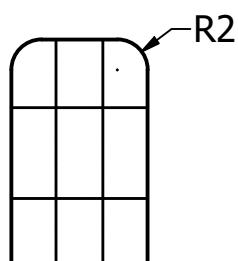
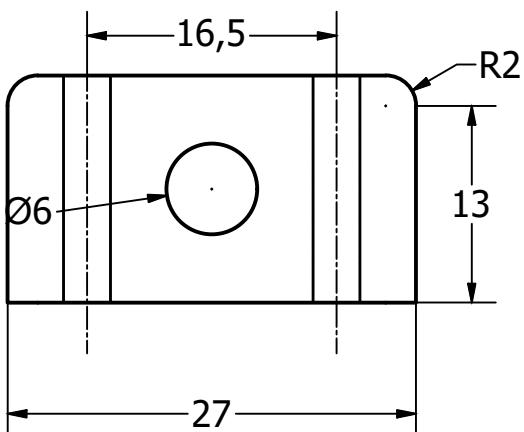
|  |   |             |
|--|---|-------------|
| INGENIERO  | PROYECTO  |             |
| Firmado por VARELA MENENDEZ DANIEL - 45960399M el día 09/09/2020 con un certificado emitido por AC FNMT Usuarios | PRÓYECTO: Clasificación de residuos mediante aprendizaje automático, con gestión centralizada de datos a través de comunicaciones de largo alcance y bajo consumo |             |
| Daniel Varela Menéndez   | PETICIONARIO: E.E.I. Universidad de Vigo  |             |
| REFERENCIA   | PLANO   | ESCALA      |
| P20  | Sujección ESP32   | 2:1         |
| FECHA  |   | Nº DE PLANO |
| 19/08/20   |   | 20/26       |

Perspectiva  
(1:1)

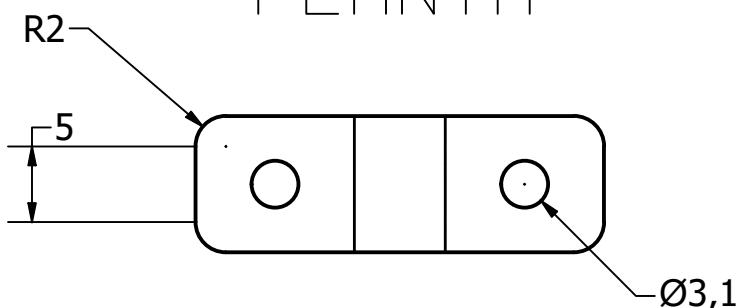


ALZADO

PERFIL IZQ.



PLANTA



## INGENIERO

Firmado por VARELA  
MENENDEZ DANIEL -  
45960399M el día  
09/09/2020 con un  
certificado emitido  
por AC FNMT Usuarios

Daniel Varela Menéndez

## REFERENCIA

P21

## FECHA

19/08/20

## PROYECTO

PROYECTO: Clasificación de residuos  
mediante aprendizaje automático,  
con gestión centralizada de datos  
a través de comunicaciones  
de largo alcance y bajo consumo  
PETICIONARIO: E.E.I. Universidad de Vigo

## PLANO

Sujección antena

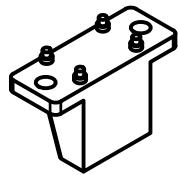


## ESCALA

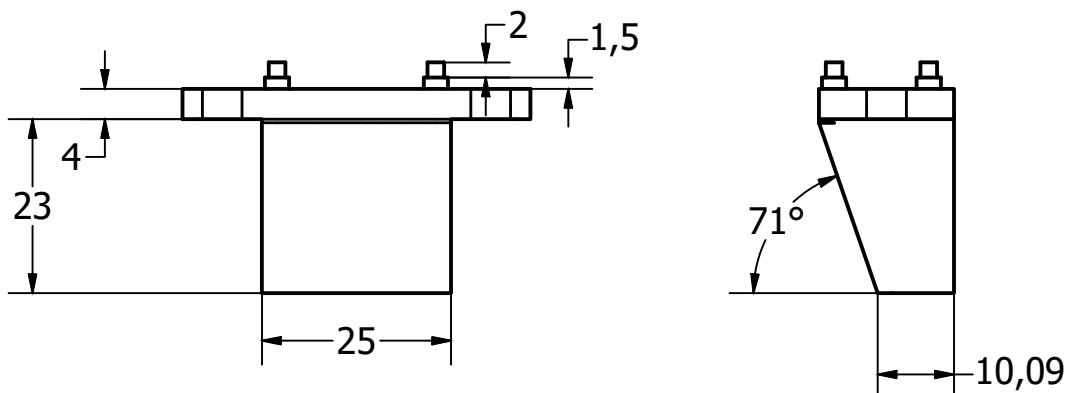
2:1

## Nº DE PLANO

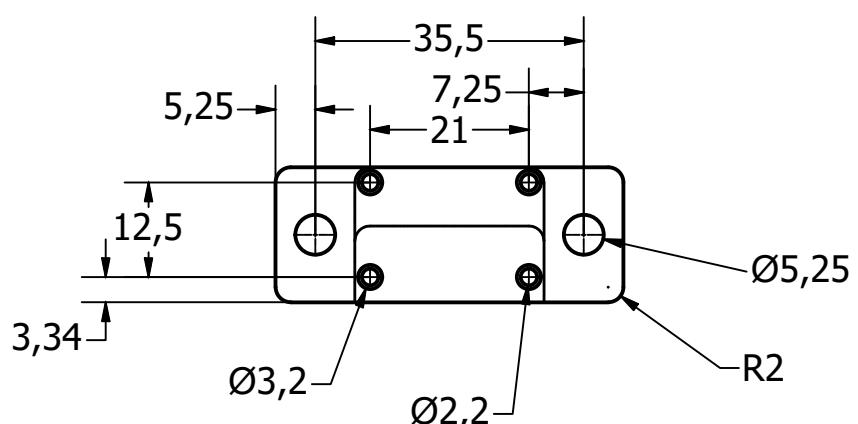
21/26

Perspectiva  
(1:2)

## ALZADO      PERFIL IZQ.



## PLANTA



## INGENIERO

Firmado por VARELA  
MENENDEZ DANIEL -  
45960399M el día  
09/09/2020 con un  
certificado emitido  
por AC FNMT Usuarios

Daniel Varela Menéndez

## REFERENCIA

P22

## FECHA

19/08/20

## PROYECTO

PROYECTO: Clasificación de residuos  
mediante aprendizaje automático,  
con gestión centralizada de datos  
a través de comunicaciones  
de largo alcance y bajo consumo  
PETICIONARIO: E.E.I. Universidad de Vigo



## PLANO

Sujeción cámara

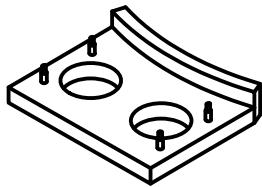
## ESCALA

1:1

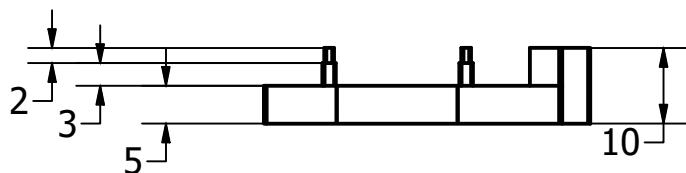
## Nº DE PLANO

22/26

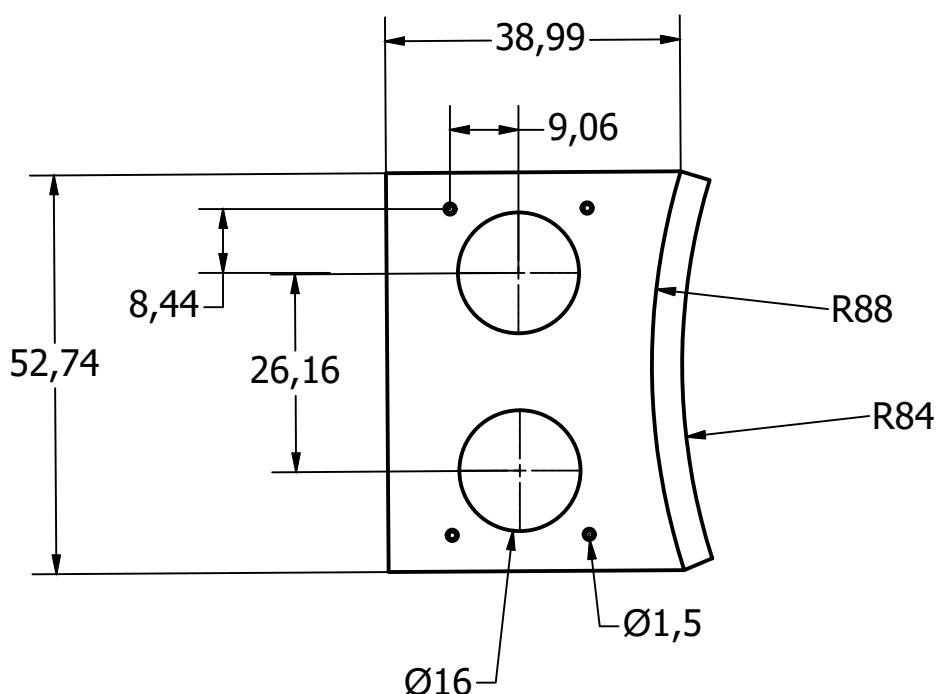
Perspectiva  
(1:2)



## ALZADO □

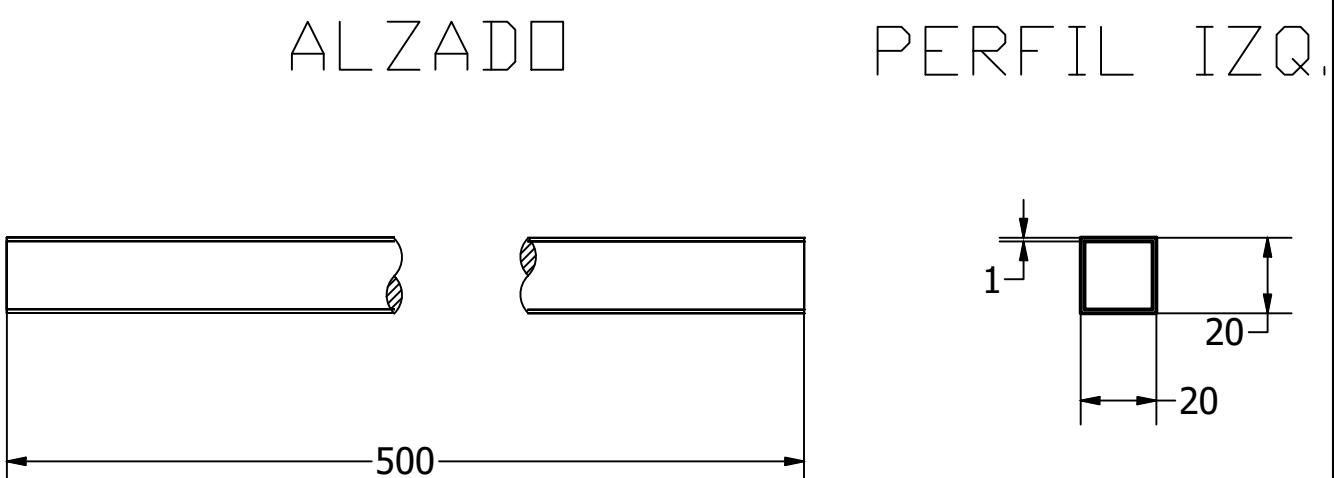


## PLANTA



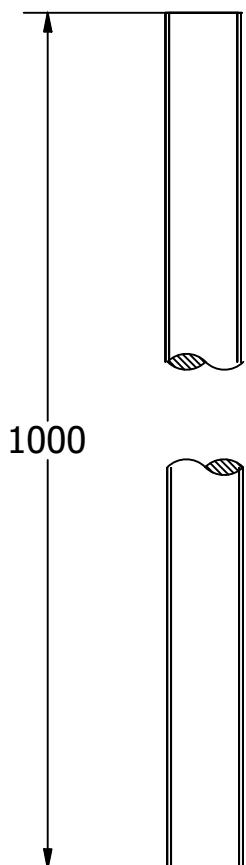
|  |   |             |
|--|---|-------------|
| INGENIERO  | PROYECTO  | ESCALA      |
| Firmado por VARELA MENENDEZ DANIEL - 45960399M el día 09/09/2020 con un certificado emitido por AC FNMT Usuarios | PRÓYECTO: Clasificación de residuos mediante aprendizaje automático, con gestión centralizada de datos a través de comunicaciones de largo alcance y bajo consumo<br>PETICIONARIO: E.E.I. Universidad de Vigo | 1:1         |
| Daniel Varela Menéndez   |   | Nº DE PLANO |
| REFERENCIA   | PLANO   | 23/26       |
| P23  | Sujeción ultrasonidos   |             |
| FECHA  |   |             |
| 19/08/20   |   |             |



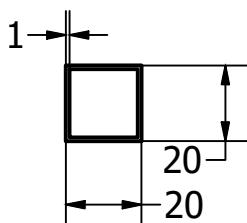


|   |   |                   |
|---|---|-------------------|
| INGENIERO   | PROYECTO  | ESCALA            |
| Firmado por VARELA<br>MENENDEZ DANIEL -<br>45960399M el día<br>09/09/2020 con un<br>certificado emitido<br>por AC FNMT Usuarios | PRÓYECTO: Clasificación de residuos<br>mediante aprendizaje automático,<br>con gestión centralizada de datos<br>a través de comunicaciones<br>de largo alcance y bajo consumo<br>PETICIONARIO: E.E.I. Universidad de Vigo | Universida deVigo |
| REFERENCIA  | PLANO   | Nº DE PLANO       |
| P24   | Tubo horizontal   | 1:2               |
| FECHA   |   | 24/26             |
| 19/08/20  |   |                   |

## ALZADO



## PLANTA



## INGENIERO

Firmado por VARELA  
MENENDEZ DANIEL -  
45960399M el día  
09/09/2020 con un  
certificado emitido  
por AC FNMT Usuarios

Daniel Varela Menéndez

## REFERENCIA

P25

## FECHA

19/08/20

## PROYECTO

PROYECTO: Clasificación de residuos  
mediante aprendizaje automático,  
con gestión centralizada de datos  
a través de comunicaciones  
de largo alcance y bajo consumo  
PETICIONARIO: E.E.I. Universidad de Vigo

## PLANO

Tubo vertical



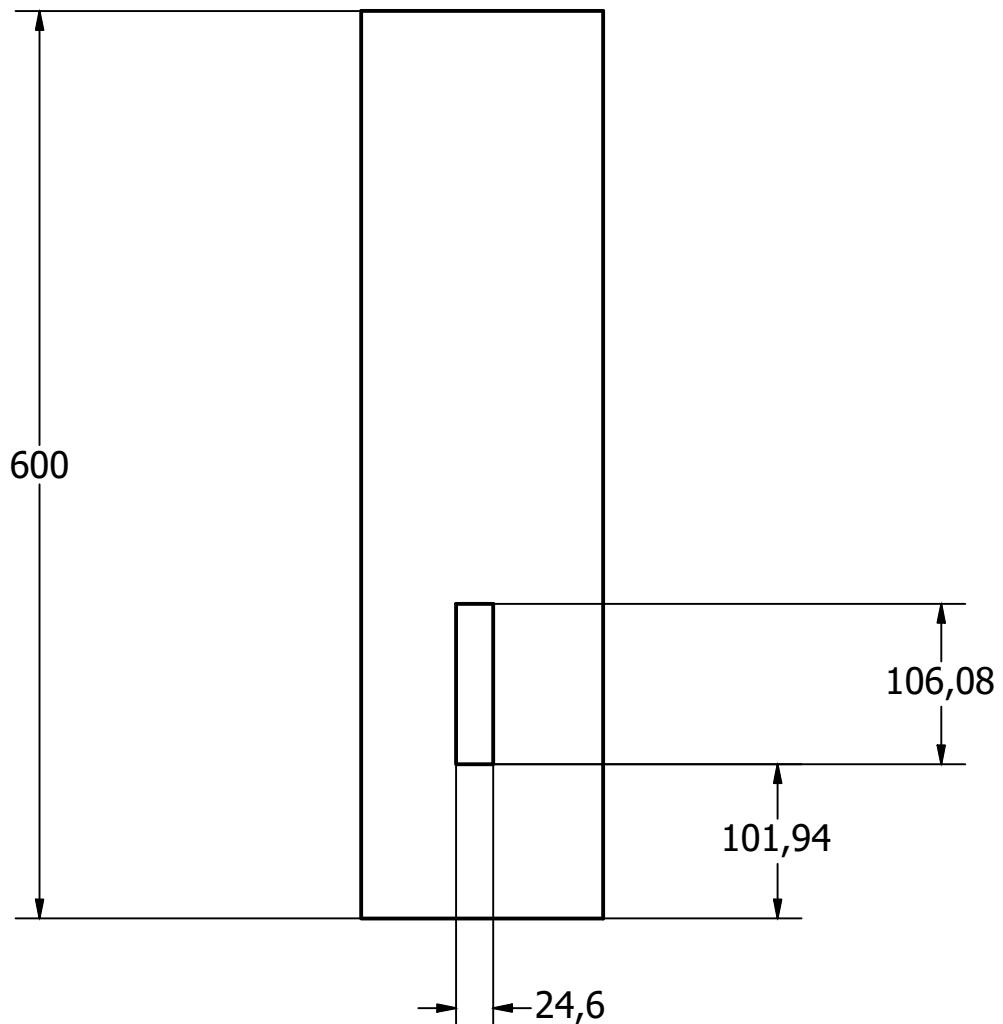
## ESCALA

1:2

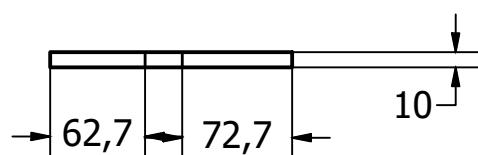
## Nº DE PLANO

25/26

## ALZADO



## PLANTA



INGENIERO  
Firmado por VARELA  
MENENDEZ DANIEL -  
45960399M el día  
09/09/2020 con un  
certificado emitido  
por AC FNMT Usuarios

Daniel Varela Menéndez

REFERENCIA

P26

FECHA

19/08/20

PROYECTO  
PRÓYECTO: Clasificación de residuos  
mediante aprendizaje automático,  
con gestión centralizada de datos  
a través de comunicaciones  
de largo alcance y bajo consumo  
PETICIONARIO: E.E.I. Universidad de Vigo

PLANO

Pared compuerta



ESCALA

1:5

Nº DE PLANO

26/26

## **15.2. Códigos de programas**

**15.2.1. *Código de la red neuronal***

```
from google.colab import drive #permite que acceda a google drive donde esta el directorio  
drive.mount('/content/drive')
```

↳ Go to this URL in a browser: [https://accounts.google.com/o/oauth2/auth?client\\_id=947](https://accounts.google.com/o/oauth2/auth?client_id=947)

Enter your authorization code:  
.....  
Mounted at /content/drive

## ▼ Importación de las librerías

```
import itertools  
import os #libreria util para realizar operaciones relacionadas con el sistema operativo  
import matplotlib.pyplot as plt #util para mostrar graficos  
import numpy as np # libreria necesaria para calculos matematicos  
import tensorflow as tf #libreria de aprendizaje automatico  
import tensorflow_hub as hub #libreria con multiples redes neuronales
```

```
print("GPU is", "available" if tf.test.is_gpu_available() else "NOT AVAILABLE") #confirma
```

↳ WARNING:tensorflow:From <ipython-input-5-606279e2e697>:8: is\_gpu\_available (from ten  
Instructions for updating:  
Use `tf.config.list\_physical\_devices('GPU')` instead.  
GPU is available

## ▼ Pre-procesamiento de las imágenes

```
MODULE_HANDLE ="https://tfhub.dev/google/imagenet/inception_v3/classification/4" #en esta  
IMAGE_SIZE = (299, 299) #la imagen que puede procesar INCEPTIONN V3 es de 299x299 pixeles,
```

```
BATCH_SIZE = 32 #el batch size contará con 32 fotografias
```

```
data_dir = '/content/drive/My Drive/SMARTBIN DVM/DATASET/TRAINING' #directorio en google c
```

```
datagen_kwargs = dict(rescale=1./255, validation_split=.20) #se establece un rescalado c  
dataflow_kwargs = dict(target_size=IMAGE_SIZE, batch_size=BATCH_SIZE, #se ajusta el tamañ  
interpolation="bilinear") # la interpolacion es bilineal
```

```
#validación  
valid_datagen = tf.keras.preprocessing.image.ImageDataGenerator(  
    **datagen_kwargs) # con los parametros anteriormente comentados se generan las imager  
valid_generator = valid_datagen.flow_from_directory(  
    data_dir, subset="validation", shuffle=False, **dataflow_kwargs) #las imagenes procesa
```

```
#entrenamiento
```

```
train_datagen = tf.keras.preprocessing.image.ImageDataGenerator(
    rotation_range=40, # se rotan aleatoriamente hasta 40º las imágenes
    horizontal_flip=True,
    width_shift_range=0.2, height_shift_range=0.2,
    shear_range=0.2, zoom_range=0.2, # se efectúa de forma aleatoria zoom sobre las imágenes
    **datagen_kwargs)
train_generator = train_datagen.flow_from_directory(
    data_dir, subset="training", shuffle=True, **dataflow_kwargs) #las imágenes procesadas
```

↳ Found 508 images belonging to 5 classes.  
Found 2041 images belonging to 5 classes.

## ▼ Construcción de la red

```
model = tf.keras.Sequential([ # se construye una red neuronal con las siguientes capas o
    tf.keras.layers.InputLayer(input_shape=IMAGE_SIZE + (3,)), # se crea una capa de entrada
    hub.KerasLayer(MODULE_HANDLE, trainable=False), # se añade INCEPTION V3 y se establece
    tf.keras.layers.Dropout(rate=0.5), #se añade una capa de Dropout con una desactivación
    tf.keras.layers.Dense(train_generator.num_classes, 'softmax', # finalmente se añade la capa de salida
        kernel_regularizer=tf.keras.regularizers.l2(0.0001)) # el regularizador
])
model.build((None,) + IMAGE_SIZE + (3,)) # se construye la red anteriormente descrita
model.summary() # se presenta un resumen de dicha red
```

↳ Model: "sequential"

| Layer (type)             | Output Shape | Param #  |
|--------------------------|--------------|----------|
| <hr/>                    |              |          |
| keras_layer (KerasLayer) | (None, 1001) | 23853833 |
| dropout (Dropout)        | (None, 1001) | 0        |
| dense (Dense)            | (None, 5)    | 5010     |
| <hr/>                    |              |          |
| Total params:            | 23,858,843   |          |
| Trainable params:        | 5,010        |          |
| Non-trainable params:    | 23,853,833   |          |

---

## ▼ Compilación

**1ºADAM+ SGD 20+80 epochas (LR=0.01 y momentum del SGD  
=0.9)**

1.1ADAM (20 epochas)

```
optimizer=tf.keras.optimizers.Adam(lr=0.01), #se empleará el optimizador Adam (Adaptive  
loss=tf.keras.losses.CategoricalCrossentropy(from_logits=True, label_smoothing=0.1), #la  
metrics=['accuracy'])  
  
epocas=20  
  
steps_per_epoch = train_generator.samples // train_generator.batch_size # para calcular  
validation_steps = valid_generator.samples // valid_generator.batch_size #idem para la eti  
hist1 = model.fit( #se ejecuta el entrenamiento  
    train_generator,  
    epochs=epocas, steps_per_epoch=steps_per_epoch, # se establecen las épocas y los pasos  
    validation_data=valid_generator,  
    validation_steps=validation_steps).history
```

Epoch 1/20  
63/63 [=====] - 834s 13s/step - loss: 1.3655 - accuracy: 0.  
Epoch 2/20  
63/63 [=====] - 65s 1s/step - loss: 1.3019 - accuracy: 0.65  
Epoch 3/20  
63/63 [=====] - 65s 1s/step - loss: 1.2893 - accuracy: 0.66  
Epoch 4/20  
63/63 [=====] - 65s 1s/step - loss: 1.2903 - accuracy: 0.66  
Epoch 5/20  
63/63 [=====] - 65s 1s/step - loss: 1.2851 - accuracy: 0.67  
Epoch 6/20  
63/63 [=====] - 65s 1s/step - loss: 1.2853 - accuracy: 0.67  
Epoch 7/20  
63/63 [=====] - 65s 1s/step - loss: 1.2853 - accuracy: 0.67  
Epoch 8/20  
63/63 [=====] - 65s 1s/step - loss: 1.2649 - accuracy: 0.69  
Epoch 9/20  
63/63 [=====] - 65s 1s/step - loss: 1.2139 - accuracy: 0.76  
Epoch 10/20  
63/63 [=====] - 65s 1s/step - loss: 1.1974 - accuracy: 0.78  
Epoch 11/20  
63/63 [=====] - 65s 1s/step - loss: 1.1935 - accuracy: 0.79  
Epoch 12/20  
63/63 [=====] - 65s 1s/step - loss: 1.2026 - accuracy: 0.78  
Epoch 13/20  
63/63 [=====] - 65s 1s/step - loss: 1.1891 - accuracy: 0.80  
Epoch 14/20  
63/63 [=====] - 66s 1s/step - loss: 1.1960 - accuracy: 0.79  
Epoch 15/20  
63/63 [=====] - 65s 1s/step - loss: 1.2039 - accuracy: 0.78  
Epoch 16/20  
63/63 [=====] - 65s 1s/step - loss: 1.2019 - accuracy: 0.79  
Epoch 17/20  
63/63 [=====] - 65s 1s/step - loss: 1.1965 - accuracy: 0.80  
Epoch 18/20  
63/63 [=====] - 66s 1s/step - loss: 1.1756 - accuracy: 0.82  
Epoch 19/20  
63/63 [=====] - 66s 1s/step - loss: 1.1912 - accuracy: 0.80  
Epoch 20/20  
63/63 [=====] - 65s 1s/step - loss: 1.1973 - accuracy: 0.80

## 1.2 SGD (80 épocas)

```
model.compile(  
    optimizer=tf.keras.optimizers.SGD(lr=0.01, momentum=0.9 ), #se emplea el optimizador SGD  
    loss=tf.keras.losses.CategoricalCrossentropy(from_logits=True, label_smoothing=0.1), #la  
    metrics=['accuracy'])  
  
epochas =80 #para esta etapa del entrenamiento se emplean 80 epochas  
  
steps_per_epoch = train_generator.samples // train_generator.batch_size  
validation_steps = valid_generator.samples // valid_generator.batch_size  
hist12 = model.fit(  
    train_generator,  
    epochs=epochas, steps_per_epoch=steps_per_epoch,  
    validation_data=valid_generator,  
    validation_steps=validation_steps,).history
```



```
Epoch 52/80
63/63 [=====] - 65s 1s/step - loss: 1.1630 - accuracy: 0.83
Epoch 53/80
63/63 [=====] - 65s 1s/step - loss: 1.1558 - accuracy: 0.84
Epoch 54/80
63/63 [=====] - 65s 1s/step - loss: 1.1584 - accuracy: 0.84
Epoch 55/80
63/63 [=====] - 65s 1s/step - loss: 1.1609 - accuracy: 0.83
Epoch 56/80
63/63 [=====] - 65s 1s/step - loss: 1.1688 - accuracy: 0.82
Epoch 57/80
63/63 [=====] - 65s 1s/step - loss: 1.1711 - accuracy: 0.82
Epoch 58/80
63/63 [=====] - 65s 1s/step - loss: 1.1478 - accuracy: 0.85
Epoch 59/80
63/63 [=====] - 65s 1s/step - loss: 1.1487 - accuracy: 0.85
Epoch 60/80
63/63 [=====] - 65s 1s/step - loss: 1.1532 - accuracy: 0.84
Epoch 61/80
63/63 [=====] - 66s 1s/step - loss: 1.1556 - accuracy: 0.84
Epoch 62/80
63/63 [=====] - 65s 1s/step - loss: 1.1392 - accuracy: 0.86
Epoch 63/80
63/63 [=====] - 65s 1s/step - loss: 1.1619 - accuracy: 0.83
Epoch 64/80
63/63 [=====] - 65s 1s/step - loss: 1.1544 - accuracy: 0.84
Epoch 65/80
63/63 [=====] - 65s 1s/step - loss: 1.1603 - accuracy: 0.83
Epoch 66/80
63/63 [=====] - 65s 1s/step - loss: 1.1664 - accuracy: 0.82
Epoch 67/80
63/63 [=====] - 66s 1s/step - loss: 1.1618 - accuracy: 0.83
Epoch 68/80
63/63 [=====] - 65s 1s/step - loss: 1.1540 - accuracy: 0.84
Epoch 69/80
63/63 [=====] - 65s 1s/step - loss: 1.1474 - accuracy: 0.85
Epoch 70/80
63/63 [=====] - 64s 1s/step - loss: 1.1605 - accuracy: 0.83
Epoch 71/80
63/63 [=====] - 65s 1s/step - loss: 1.1537 - accuracy: 0.84
Epoch 72/80
63/63 [=====] - 64s 1s/step - loss: 1.1600 - accuracy: 0.83
Epoch 73/80
63/63 [=====] - 64s 1s/step - loss: 1.1517 - accuracy: 0.84
Epoch 74/80
63/63 [=====] - 65s 1s/step - loss: 1.1545 - accuracy: 0.84
Epoch 75/80
63/63 [=====] - 64s 1s/step - loss: 1.1634 - accuracy: 0.83
Epoch 76/80
63/63 [=====] - 65s 1s/step - loss: 1.1470 - accuracy: 0.85
Epoch 77/80
63/63 [=====] - 65s 1s/step - loss: 1.1505 - accuracy: 0.84
Epoch 78/80
63/63 [=====] - 66s 1s/step - loss: 1.1567 - accuracy: 0.84
Epoch 79/80
63/63 [=====] - 65s 1s/step - loss: 1.1563 - accuracy: 0.84
Epoch 80/80
63/63 [=====] - 66s 1s/step - loss: 1.1519 - accuracy: 0.84
```

```
model.save('content/drive/My Drive/SMARTBIN DVM/modelos/modelopb_1')
```



```
WARNING:tensorflow:From /usr/local/lib/python3.6/dist-packages/tensorflow/python/tra
Instructions for updating:
This property should not be used in TensorFlow 2.0, as updates are applied automatic
WARNING:tensorflow:From /usr/local/lib/python3.6/dist-packages/tensorflow/python/tra
Instructions for updating:
This property should not be used in TensorFlow 2.0, as updates are applied automatic
WARNING:tensorflow:From /usr/local/lib/python3.6/dist-packages/tensorflow/python/tra
Instructions for updating:
This property should not be used in TensorFlow 2.0, as updates are applied automatic
WARNING:tensorflow:From /usr/local/lib/python3.6/dist-packages/tensorflow/python/tra
Instructions for updating:
This property should not be used in TensorFlow 2.0, as updates are applied automatic
INFO:tensorflow:Assets written to: /content/drive/My Drive/SMARTBIN DVM/modelos/mode
INFO:tensorflow:Assets written to: /content/drive/My Drive/SMARTBIN DVM/modelos/mode
```

## #1.1 ADAM

```
#Visualización de la precisión del entrenamiento y de la validacion
plt.figure()
plt.ylabel("Precisión ")
plt.xlabel("Pasos")
plt.ylim([0,1])
plt.plot(hist1["accuracy"])
plt.plot(hist1["val_accuracy"])
plt.legend(['Entrenamiento','Validacion']);
plt.title ("Precisión frente a Épocas (Etapa ADAM de 20 épocas)")
```

## #Valor de la función de pérdida de la etapa de entrenamiento y validacion

```
plt.figure()
plt.ylabel("Perdida ")
plt.xlabel("Pasos ")
plt.ylim([0,2])
plt.plot(hist1["loss"])
plt.plot(hist1["val_loss"],)
plt.legend(['Entrenamiento','Validacion']);
plt.title ("Perdida frente a Épocas (Etapa ADAM de 20 épocas)")
#1.2 SGD
```

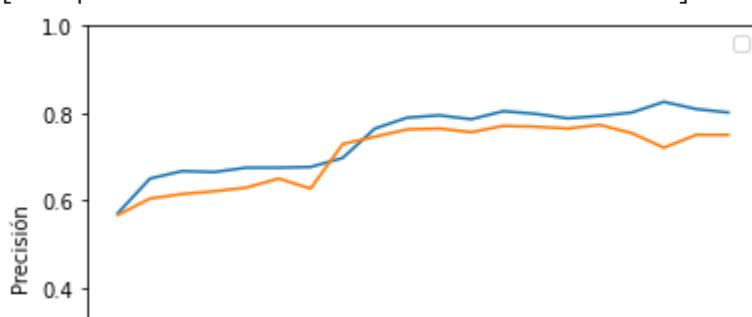
```
#Visualización de la precisión del entrenamiento y de la validacion
plt.figure()
plt.ylabel("Precisión ")
plt.xlabel("Pasos")
plt.ylim([0,1])
plt.plot(hist12["accuracy"])
plt.plot(hist12["val_accuracy"])
plt.legend(['Entrenamiento','Validacion']);
plt.title ("Precisión frente a Épocas (Etapa SGD de 80 épocas)")
```

## #Valor de la función de pérdida de la etapa de entrenamiento y validacion

```
plt.figure()
plt.ylabel("Perdida ")
plt.xlabel("Pasos ")
plt.ylim([0,2])
```

```
plt.plot(hist12["loss"])
plt.plot(hist12["val_loss"])
plt.legend(['Entrenamiento','Validacion']);
plt.title ("Perdida frente a Épocas (Etapa SGD de 80 épocas)")
```





## ▼ 2º ADAM (LR=0.01)

0.0 |

```
model2 = tf.keras.Sequential([ # se construye una red neuronal con las siguientes capas
    tf.keras.layers.InputLayer(input_shape=IMAGE_SIZE + (3,)), # se crea una capa de entrada
    hub.KerasLayer(MODULE_HANDLE, trainable=False), # se añade INCEPTION V3 y se establece
    tf.keras.layers.Dropout(rate=0.5), # se añade una capa de Dropout con una desactivación
    tf.keras.layers.Dense(train_generator.num_classes, 'softmax', # finalmente se añade la capa de salida
        kernel_regularizer=tf.keras.regularizers.l2(0.0001)) # el regularizador es l2(0.0001)
])
model2.build((None,) + IMAGE_SIZE + (3,)) # se construye la red anteriormente descrita
model2.summary() # se presenta un resumen de dicha red
```

⇨ Model: "sequential\_1"

| Layer (type)               | Output Shape | Param #  |
|----------------------------|--------------|----------|
| <hr/>                      |              |          |
| keras_layer_1 (KerasLayer) | (None, 1001) | 23853833 |
| dropout_1 (Dropout)        | (None, 1001) | 0        |
| dense_1 (Dense)            | (None, 5)    | 5010     |
| <hr/>                      |              |          |
| Total params:              | 23,858,843   |          |
| Trainable params:          | 5,010        |          |
| Non-trainable params:      | 23,853,833   |          |

---

| |

```
model2.compile( # se compila el modelo
    optimizer=tf.keras.optimizers.Adam(
        learning_rate=0.01, beta_1=0.9, beta_2=0.999, epsilon=1e-07, amsgrad=False,
        name='Adam'
    ), # se empleará el optimizador Adam (Adaptive Moment Estimation)
    loss=tf.keras.losses.CategoricalCrossentropy(from_logits=True, label_smoothing=0.1), # la función de pérdida es Categorical Cross Entropy
    metrics=['accuracy'])
```

↑↑↑ | |

epochas = 100

```
steps_per_epoch = train_generator.samples // train_generator.batch_size
validation_steps = valid_generator.samples // valid_generator.batch_size
hist2 = model2.fit(
    train_generator,
    epochs=epochas, steps_per_epoch=steps_per_epoch,
    validation_data=valid_generator,
    validation_steps=validation_steps).history
```

⇨

```
Epoch 72/100
63/63 [=====] - 65s 1s/step - loss: 1.1994 - accuracy: 0.81
Epoch 73/100
63/63 [=====] - 65s 1s/step - loss: 1.2106 - accuracy: 0.80
Epoch 74/100
63/63 [=====] - 66s 1s/step - loss: 1.1915 - accuracy: 0.82
Epoch 75/100
63/63 [=====] - 65s 1s/step - loss: 1.1971 - accuracy: 0.81
Epoch 76/100
63/63 [=====] - 66s 1s/step - loss: 1.2006 - accuracy: 0.81
Epoch 77/100
63/63 [=====] - 66s 1s/step - loss: 1.2116 - accuracy: 0.80
Epoch 78/100
63/63 [=====] - 66s 1s/step - loss: 1.2041 - accuracy: 0.81
Epoch 79/100
63/63 [=====] - 67s 1s/step - loss: 1.2198 - accuracy: 0.79
Epoch 80/100
63/63 [=====] - 66s 1s/step - loss: 1.2001 - accuracy: 0.81
Epoch 81/100
63/63 [=====] - 66s 1s/step - loss: 1.1956 - accuracy: 0.82
Epoch 82/100
63/63 [=====] - 66s 1s/step - loss: 1.2002 - accuracy: 0.81
Epoch 83/100
63/63 [=====] - 65s 1s/step - loss: 1.2027 - accuracy: 0.81
Epoch 84/100
63/63 [=====] - 65s 1s/step - loss: 1.2028 - accuracy: 0.81
Epoch 85/100
63/63 [=====] - 66s 1s/step - loss: 1.2004 - accuracy: 0.81
Epoch 86/100
63/63 [=====] - 66s 1s/step - loss: 1.1954 - accuracy: 0.81
Epoch 87/100
63/63 [=====] - 66s 1s/step - loss: 1.1939 - accuracy: 0.82
Epoch 88/100
63/63 [=====] - 65s 1s/step - loss: 1.1948 - accuracy: 0.82
Epoch 89/100
63/63 [=====] - 65s 1s/step - loss: 1.2000 - accuracy: 0.81
Epoch 90/100
63/63 [=====] - 65s 1s/step - loss: 1.2017 - accuracy: 0.81
Epoch 91/100
63/63 [=====] - 65s 1s/step - loss: 1.2067 - accuracy: 0.81
Epoch 92/100
63/63 [=====] - 65s 1s/step - loss: 1.2127 - accuracy: 0.80
Epoch 93/100
63/63 [=====] - 65s 1s/step - loss: 1.1955 - accuracy: 0.82
Epoch 94/100
63/63 [=====] - 65s 1s/step - loss: 1.2070 - accuracy: 0.81
Epoch 95/100
63/63 [=====] - 65s 1s/step - loss: 1.2072 - accuracy: 0.81
Epoch 96/100
63/63 [=====] - 66s 1s/step - loss: 1.2014 - accuracy: 0.81
Epoch 97/100
63/63 [=====] - 66s 1s/step - loss: 1.1992 - accuracy: 0.82
Epoch 98/100
63/63 [=====] - 65s 1s/step - loss: 1.1982 - accuracy: 0.82
Epoch 99/100
63/63 [=====] - 66s 1s/step - loss: 1.2013 - accuracy: 0.81
Epoch 100/100
63/63 [=====] - 66s 1s/step - loss: 1.2121 - accuracy: 0.80
```





```
model2.save('/content/drive/My Drive/SMARTBIN DVM/modelos/modelopb_2')
```

↳ INFO:tensorflow:Assets written to: /content/drive/My Drive/SMARTBIN DVM/modelos/mode  
INFO:tensorflow:Assets written to: /content/drive/My Drive/SMARTBIN DVM/modelos/mode

```
#Visualización de la precisión del entrenamiento y de la validacion
```

```
plt.figure()  
plt.ylabel("Precisión ")  
plt.xlabel("Pasos")  
plt.ylim([0,1])  
plt.plot(hist2["accuracy"])  
plt.plot(hist2["val_accuracy"]);  
plt.legend(['Entrenamiento','Validacion']);  
plt.title ("Precisión frente a Épocas")
```

```
#Valor de la función de pérdida de la etapa de entrenamiento y validacion
```

```
plt.figure()  
plt.ylabel("Perdida ")  
plt.xlabel("Pasos ")  
plt.ylim([0,2])  
plt.plot(hist2["loss"])  
plt.plot(hist2["val_loss"],)  
plt.legend(['Entrenamiento','Validacion']);  
plt.title ("Perdida frente a Épocas")
```

↳

/usr/local/lib/python3.6/dist-packages/ipykernel\_launcher.py:8: UserWarning: Legend  
A proxy artist may be used instead.

See: [http://matplotlib.org/users/legend\\_guide.html#creating-artists-specifically-for](http://matplotlib.org/users/legend_guide.html#creating-artists-specifically-for)

/usr/local/lib/python3.6/dist-packages/ipykernel\_launcher.py:8: UserWarning: Legend  
A proxy artist may be used instead.

See: [http://matplotlib.org/users/legend\\_guide.html#creating-artists-specifically-for](http://matplotlib.org/users/legend_guide.html#creating-artists-specifically-for)

/usr/local/lib/python3.6/dist-packages/ipykernel\_launcher.py:8: UserWarning: Legend  
A proxy artist may be used instead.

See: [http://matplotlib.org/users/legend\\_guide.html#creating-artists-specifically-for](http://matplotlib.org/users/legend_guide.html#creating-artists-specifically-for)

/usr/local/lib/python3.6/dist-packages/ipykernel\_launcher.py:8: UserWarning: Legend  
A proxy artist may be used instead.

See: [http://matplotlib.org/users/legend\\_guide.html#creating-artists-specifically-for](http://matplotlib.org/users/legend_guide.html#creating-artists-specifically-for)

/usr/local/lib/python3.6/dist-packages/ipykernel\_launcher.py:8: UserWarning: Legend  
A proxy artist may be used instead.

See: [http://matplotlib.org/users/legend\\_guide.html#creating-artists-specifically-for](http://matplotlib.org/users/legend_guide.html#creating-artists-specifically-for)

/usr/local/lib/python3.6/dist-packages/ipykernel\_launcher.py:8: UserWarning: Legend  
A proxy artist may be used instead.

See: [http://matplotlib.org/users/legend\\_guide.html#creating-artists-specifically-for](http://matplotlib.org/users/legend_guide.html#creating-artists-specifically-for)

/usr/local/lib/python3.6/dist-packages/ipykernel\_launcher.py:8: UserWarning: Legend  
A proxy artist may be used instead.

See: [http://matplotlib.org/users/legend\\_guide.html#creating-artists-specifically-for](http://matplotlib.org/users/legend_guide.html#creating-artists-specifically-for)

/usr/local/lib/python3.6/dist-packages/ipykernel\_launcher.py:8: UserWarning: Legend  
A proxy artist may be used instead.

See: [http://matplotlib.org/users/legend\\_guide.html#creating-artists-specifically-for](http://matplotlib.org/users/legend_guide.html#creating-artists-specifically-for)

/usr/local/lib/python3.6/dist-packages/ipykernel\_launcher.py:8: UserWarning: Legend  
A proxy artist may be used instead.

See: [http://matplotlib.org/users/legend\\_guide.html#creating-artists-specifically-for](http://matplotlib.org/users/legend_guide.html#creating-artists-specifically-for)

/usr/local/lib/python3.6/dist-packages/ipykernel\_launcher.py:8: UserWarning: Legend  
A proxy artist may be used instead.

See: [http://matplotlib.org/users/legend\\_guide.html#creating-artists-specifically-for](http://matplotlib.org/users/legend_guide.html#creating-artists-specifically-for)

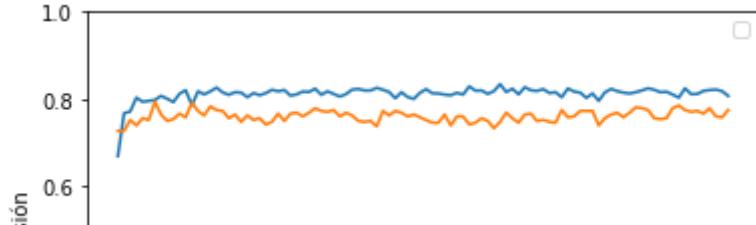
/usr/local/lib/python3.6/dist-packages/ipykernel\_launcher.py:8: UserWarning: Legend  
A proxy artist may be used instead.

See: [http://matplotlib.org/users/legend\\_guide.html#creating-artists-specifically-for](http://matplotlib.org/users/legend_guide.html#creating-artists-specifically-for)

/usr/local/lib/python3.6/dist-packages/ipykernel\_launcher.py:8: UserWarning: Legend  
A proxy artist may be used instead.

See: [http://matplotlib.org/users/legend\\_guide.html#creating-artists-specifically-for](http://matplotlib.org/users/legend_guide.html#creating-artists-specifically-for)

[<matplotlib.lines.Line2D at 0x7f1c9397e198>]



## ▼ 3°SGD (LR=0.01)

0.2 ↴

|

```
model3 = tf.keras.Sequential([
    # se construye una red neuronal con las siguientes capas
    tf.keras.layers.InputLayer(input_shape=IMAGE_SIZE + (3,)), # se crea una capa de entrada
    hub.KerasLayer(MODULE_HANDLE, trainable=False), # se añade INCEPTION V3 y se establece
    tf.keras.layers.Dropout(rate=0.5), # se añade una capa de Dropout con una desactivación
    tf.keras.layers.Dense(train_generator.num_classes, 'softmax', # finalmente se añade la capa de salida
        kernel_regularizer=tf.keras.regularizers.l2(0.0001)) # el regularizador es l2(0.0001)
])
model3.build((None,) + IMAGE_SIZE + (3,)) # se construye la red anteriormente descrita
model3.summary() # se presenta un resumen de dicha red
```

↳ Model: "sequential"

| Layer (type)             | Output Shape | Param #  |
|--------------------------|--------------|----------|
| <hr/>                    |              |          |
| keras_layer (KerasLayer) | (None, 1001) | 23853833 |
| dropout (Dropout)        | (None, 1001) | 0        |
| dense (Dense)            | (None, 5)    | 5010     |
| <hr/>                    |              |          |
| Total params:            | 23,858,843   |          |
| Trainable params:        | 5,010        |          |
| Non-trainable params:    | 23,853,833   |          |

---

```
model3.compile(
    optimizer=tf.keras.optimizers.SGD(
        learning_rate=0.01, momentum=0.9, nesterov=False, name='SGD'
    ), # se emplea el optimizador SGD (Stochastic Gradient Descent) con un learning rate de 0.01
    loss=tf.keras.losses.CategoricalCrossentropy(from_logits=True, label_smoothing=0.1), # la loss es CategoricalCrossentropy
    metrics=['accuracy'])
```

epochas = 100

```
steps_per_epoch = train_generator.samples // train_generator.batch_size
validation_steps = valid_generator.samples // valid_generator.batch_size
hist3 = model3.fit(
    train_generator,
    epochs=epochas, steps_per_epoch=steps_per_epoch,
    validation_data=valid_generator,
    validation_steps=validation_steps).history
```

↳

```
Epoch 1/100
63/63 [=====] - 67s 1s/step - loss: 1.3920 - accuracy: 0.53
Epoch 2/100
63/63 [=====] - 67s 1s/step - loss: 1.2148 - accuracy: 0.73
Epoch 3/100
63/63 [=====] - 67s 1s/step - loss: 1.1862 - accuracy: 0.77
Epoch 4/100
63/63 [=====] - 67s 1s/step - loss: 1.1823 - accuracy: 0.77
Epoch 5/100
63/63 [=====] - 67s 1s/step - loss: 1.1758 - accuracy: 0.79
Epoch 6/100
63/63 [=====] - 67s 1s/step - loss: 1.1613 - accuracy: 0.80
Epoch 7/100
63/63 [=====] - 66s 1s/step - loss: 1.1647 - accuracy: 0.79
Epoch 8/100
63/63 [=====] - 66s 1s/step - loss: 1.1588 - accuracy: 0.80
Epoch 9/100
63/63 [=====] - 66s 1s/step - loss: 1.1469 - accuracy: 0.82
Epoch 10/100
63/63 [=====] - 67s 1s/step - loss: 1.1472 - accuracy: 0.82
Epoch 11/100
63/63 [=====] - 67s 1s/step - loss: 1.1520 - accuracy: 0.81
Epoch 12/100
63/63 [=====] - 67s 1s/step - loss: 1.1388 - accuracy: 0.83
Epoch 13/100
63/63 [=====] - 66s 1s/step - loss: 1.1452 - accuracy: 0.82
Epoch 14/100
63/63 [=====] - 66s 1s/step - loss: 1.1411 - accuracy: 0.82
Epoch 15/100
63/63 [=====] - 67s 1s/step - loss: 1.1363 - accuracy: 0.83
Epoch 16/100
63/63 [=====] - 66s 1s/step - loss: 1.1368 - accuracy: 0.83
Epoch 17/100
63/63 [=====] - 67s 1s/step - loss: 1.1396 - accuracy: 0.82
Epoch 18/100
63/63 [=====] - 66s 1s/step - loss: 1.1337 - accuracy: 0.83
Epoch 19/100
63/63 [=====] - 66s 1s/step - loss: 1.1337 - accuracy: 0.83
Epoch 20/100
63/63 [=====] - 67s 1s/step - loss: 1.1294 - accuracy: 0.83
Epoch 21/100
63/63 [=====] - 66s 1s/step - loss: 1.1301 - accuracy: 0.84
Epoch 22/100
63/63 [=====] - 66s 1s/step - loss: 1.1262 - accuracy: 0.84
Epoch 23/100
63/63 [=====] - 66s 1s/step - loss: 1.1267 - accuracy: 0.84
Epoch 24/100
63/63 [=====] - 66s 1s/step - loss: 1.1314 - accuracy: 0.83
Epoch 25/100
63/63 [=====] - 64s 1s/step - loss: 1.1310 - accuracy: 0.84
Epoch 26/100
63/63 [=====] - 65s 1s/step - loss: 1.1279 - accuracy: 0.84
Epoch 27/100
63/63 [=====] - 65s 1s/step - loss: 1.1263 - accuracy: 0.84
Epoch 28/100
63/63 [=====] - 64s 1s/step - loss: 1.1259 - accuracy: 0.84
Epoch 29/100
63/63 [=====] - 65s 1s/step - loss: 1.1285 - accuracy: 0.84
Epoch 30/100
63/63 [=====] - 65s 1s/step - loss: 1.1207 - accuracy: 0.85
Epoch 31/100
```

```
63/63 [=====] - 65s 1s/step - loss: 1.1324 - accuracy: 0.83
Epoch 32/100
63/63 [=====] - 65s 1s/step - loss: 1.1258 - accuracy: 0.84
Epoch 33/100
63/63 [=====] - 66s 1s/step - loss: 1.1145 - accuracy: 0.86
Epoch 34/100
63/63 [=====] - 66s 1s/step - loss: 1.1215 - accuracy: 0.84
Epoch 35/100
63/63 [=====] - 66s 1s/step - loss: 1.1267 - accuracy: 0.84
Epoch 36/100
63/63 [=====] - 65s 1s/step - loss: 1.1294 - accuracy: 0.84
Epoch 37/100
63/63 [=====] - 65s 1s/step - loss: 1.1240 - accuracy: 0.85
Epoch 38/100
63/63 [=====] - 67s 1s/step - loss: 1.1242 - accuracy: 0.84
Epoch 39/100
63/63 [=====] - 66s 1s/step - loss: 1.1202 - accuracy: 0.85
Epoch 40/100
63/63 [=====] - 66s 1s/step - loss: 1.1181 - accuracy: 0.85
Epoch 41/100
63/63 [=====] - 67s 1s/step - loss: 1.1241 - accuracy: 0.84
Epoch 42/100
63/63 [=====] - 67s 1s/step - loss: 1.1191 - accuracy: 0.85
Epoch 43/100
63/63 [=====] - 67s 1s/step - loss: 1.1246 - accuracy: 0.84
Epoch 44/100
63/63 [=====] - 66s 1s/step - loss: 1.1258 - accuracy: 0.84
Epoch 45/100
63/63 [=====] - 66s 1s/step - loss: 1.1151 - accuracy: 0.85
Epoch 46/100
63/63 [=====] - 67s 1s/step - loss: 1.1142 - accuracy: 0.85
Epoch 47/100
63/63 [=====] - 67s 1s/step - loss: 1.1161 - accuracy: 0.85
Epoch 48/100
63/63 [=====] - 66s 1s/step - loss: 1.1303 - accuracy: 0.83
Epoch 49/100
63/63 [=====] - 66s 1s/step - loss: 1.1165 - accuracy: 0.85
Epoch 50/100
63/63 [=====] - 66s 1s/step - loss: 1.1258 - accuracy: 0.84
Epoch 51/100
63/63 [=====] - 66s 1s/step - loss: 1.1226 - accuracy: 0.84
Epoch 52/100
63/63 [=====] - 67s 1s/step - loss: 1.1223 - accuracy: 0.84
Epoch 53/100
63/63 [=====] - 66s 1s/step - loss: 1.1191 - accuracy: 0.85
Epoch 54/100
63/63 [=====] - 66s 1s/step - loss: 1.1165 - accuracy: 0.85
Epoch 55/100
63/63 [=====] - 66s 1s/step - loss: 1.1216 - accuracy: 0.85
Epoch 56/100
63/63 [=====] - 66s 1s/step - loss: 1.1143 - accuracy: 0.85
Epoch 57/100
63/63 [=====] - 66s 1s/step - loss: 1.1213 - accuracy: 0.85
Epoch 58/100
63/63 [=====] - 66s 1s/step - loss: 1.1180 - accuracy: 0.85
Epoch 59/100
63/63 [=====] - 67s 1s/step - loss: 1.1247 - accuracy: 0.84
Epoch 60/100
63/63 [=====] - 66s 1s/step - loss: 1.1119 - accuracy: 0.86
Epoch 61/100
63/63 [=====] - 67s 1s/step - loss: 1.1171 - accuracy: 0.85
Epoch 62/100
```

```
[=====] - 67s 1s/step - loss: 1.1103 - accuracy: 0.86
Epoch 63/100
[=====] - 67s 1s/step - loss: 1.1202 - accuracy: 0.85
Epoch 64/100
[=====] - 65s 1s/step - loss: 1.1170 - accuracy: 0.85
Epoch 65/100
[=====] - 66s 1s/step - loss: 1.1151 - accuracy: 0.85
Epoch 66/100
[=====] - 66s 1s/step - loss: 1.1186 - accuracy: 0.85
Epoch 67/100
[=====] - 65s 1s/step - loss: 1.1183 - accuracy: 0.85
Epoch 68/100
[=====] - 65s 1s/step - loss: 1.1175 - accuracy: 0.85
Epoch 69/100
[=====] - 64s 1s/step - loss: 1.1158 - accuracy: 0.85
Epoch 70/100
[=====] - 65s 1s/step - loss: 1.1150 - accuracy: 0.85
Epoch 71/100
[=====] - 65s 1s/step - loss: 1.1127 - accuracy: 0.86
Epoch 72/100
[=====] - 65s 1s/step - loss: 1.1088 - accuracy: 0.86
Epoch 73/100
[=====] - 65s 1s/step - loss: 1.1189 - accuracy: 0.85
Epoch 74/100
[=====] - 65s 1s/step - loss: 1.1303 - accuracy: 0.84
Epoch 75/100
[=====] - 67s 1s/step - loss: 1.1119 - accuracy: 0.86
Epoch 76/100
[=====] - 65s 1s/step - loss: 1.1165 - accuracy: 0.85
Epoch 77/100
[=====] - 65s 1s/step - loss: 1.1097 - accuracy: 0.86
Epoch 78/100
[=====] - 64s 1s/step - loss: 1.1210 - accuracy: 0.85
Epoch 79/100
[=====] - 65s 1s/step - loss: 1.1159 - accuracy: 0.85
Epoch 80/100
[=====] - 65s 1s/step - loss: 1.1169 - accuracy: 0.85
Epoch 81/100
[=====] - 66s 1s/step - loss: 1.1164 - accuracy: 0.86
Epoch 82/100
[=====] - 66s 1s/step - loss: 1.1176 - accuracy: 0.85
Epoch 83/100
[=====] - 66s 1s/step - loss: 1.1190 - accuracy: 0.85
Epoch 84/100
[=====] - 67s 1s/step - loss: 1.1083 - accuracy: 0.86
Epoch 85/100
[=====] - 66s 1s/step - loss: 1.1196 - accuracy: 0.85
Epoch 86/100
[=====] - 66s 1s/step - loss: 1.1151 - accuracy: 0.85
Epoch 87/100
[=====] - 64s 1s/step - loss: 1.1142 - accuracy: 0.86
Epoch 88/100
[=====] - 65s 1s/step - loss: 1.1142 - accuracy: 0.86
Epoch 89/100
[=====] - 65s 1s/step - loss: 1.1217 - accuracy: 0.85
Epoch 90/100
[=====] - 64s 1s/step - loss: 1.1116 - accuracy: 0.86
Epoch 91/100
[=====] - 64s 1s/step - loss: 1.1142 - accuracy: 0.86
Epoch 92/100
[=====] - 64s 1s/step - loss: 1.1173 - accuracy: 0.85
```

Epoch 93/100

63/63 [=====] - 65s 1s/step - loss: 1.1150 - accuracy: 0.85

```
model3.save('/content/drive/My Drive/SMARTBIN DVM/modelos/modelopb_3')
```

```
INFO:tensorflow:Assets written to: /content/drive/My Drive/SMARTBIN DVM/modelos/mode  
INFO:tensorflow:Assets written to: /content/drive/My Drive/SMARTBIN DVM/modelos/mode  
.....
```

Spec. No. 100  
CP 625

## #Visualizaci

```
plt.figure()
```

```
plt.ylabel("Precision")
```

```
plt.xlabel("Pas")
```

```
plt.ylim([0,1])
```

```
plt.plot(hist3["accuracy"])
```

```
plt.plot(hist3["val_accuracy"])
```

```
plt.legend(['Entrenamiento', 'Validacion'])
```

#Valor de la función de perdida de la etapa de entrenamiento y validación

```
plt.figure()
```

```
plt.ylabel("Perdida")
```

```
plt.xlabel("Pasos ")
```

```
plt.ylim([0,2])
```

```
plt.plot(hist3["loss"])
```

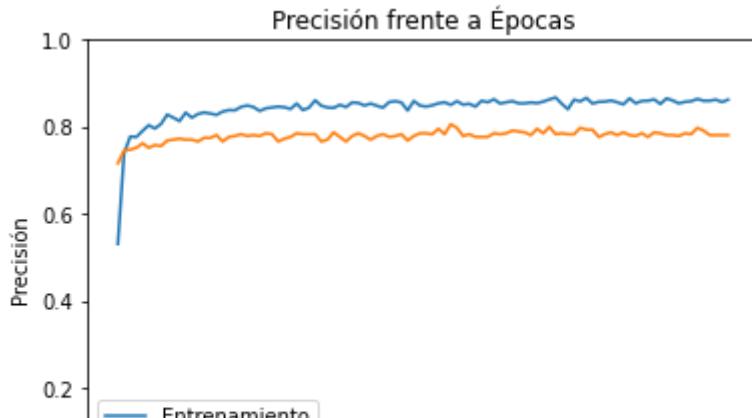
```
plt.plot(hist3["val_loss"],)
```

```
plt.legend(['Entrenamiento','Validacion']);
```

```
plt.title ("Perdida frente a Épocas")
```

1

Text(0.5, 1.0, 'Perdida frente a Épocas')



## ▼ 4º NADAM (LR=0.01)

Perdida frente a Épocas

```
model4 = tf.keras.Sequential([
    # se construye una red neuronal con las siguientes capas
    tf.keras.layers.InputLayer(input_shape=IMAGE_SIZE + (3,)), # se crea una capa de entrada
    hub.KerasLayer(MODULE_HANDLE, trainable=False), # se añade INCEPTION V3 y se establece la arquitectura
    tf.keras.layers.Dropout(rate=0.5), #se añade una capa de Dropout con una desactivación
    tf.keras.layers.Dense(train_generator.num_classes, 'softmax', # finalmente se añade la capa de salida
        kernel_regularizer=tf.keras.regularizers.l2(0.0001)) # el regularizador es l2(0.0001)
])
model4.build((None,) + IMAGE_SIZE + (3,)) # se construye la red anteriormente descrita
model4.summary() # se presenta un resumen de dicha red
```

Model: "sequential\_4"

| Layer (type)               | Output Shape | Param #  |
|----------------------------|--------------|----------|
| <hr/>                      |              |          |
| keras_layer_4 (KerasLayer) | (None, 1001) | 23853833 |
| <hr/>                      |              |          |
| dropout_4 (Dropout)        | (None, 1001) | 0        |
| <hr/>                      |              |          |
| dense_4 (Dense)            | (None, 5)    | 5010     |
| <hr/>                      |              |          |
| Total params:              | 23,858,843   |          |
| Trainable params:          | 5,010        |          |
| Non-trainable params:      | 23,853,833   |          |

---

```
model4.compile(
    optimizer=tf.keras.optimizers.Nadam(
        learning_rate=0.01, beta_1=0.9, beta_2=0.999, epsilon=1e-07, name='Nadam'
    ), #se emplea el optimizador SGD (Stochastic Gradient Descent) con un learning rate de 0.01
    loss=tf.keras.losses.CategoricalCrossentropy(from_logits=True, label_smoothing=0.1),
    metrics=['accuracy'])
```

epochas =100

```
steps_per_epoch = train_generator.samples // train_generator.batch_size
validation_steps = valid_generator.samples // valid_generator.batch_size
hist4 = model4.fit(
```

```
train_generator,  
epochs=epochas, steps_per_epoch=steps_per_epoch,  
validation_data=valid_generator,  
validation_steps=validation_steps,).history
```



```
Epoch 1/100
63/63 [=====] - 66s 1s/step - loss: 1.2692 - accuracy: 0.67
Epoch 2/100
63/63 [=====] - 65s 1s/step - loss: 1.1900 - accuracy: 0.77
Epoch 3/100
63/63 [=====] - 66s 1s/step - loss: 1.1832 - accuracy: 0.78
Epoch 4/100
63/63 [=====] - 66s 1s/step - loss: 1.1711 - accuracy: 0.79
Epoch 5/100
63/63 [=====] - 67s 1s/step - loss: 1.1843 - accuracy: 0.78
Epoch 6/100
63/63 [=====] - 66s 1s/step - loss: 1.1733 - accuracy: 0.79
Epoch 7/100
63/63 [=====] - 65s 1s/step - loss: 1.1692 - accuracy: 0.81
Epoch 8/100
63/63 [=====] - 66s 1s/step - loss: 1.1716 - accuracy: 0.80
Epoch 9/100
63/63 [=====] - 66s 1s/step - loss: 1.1699 - accuracy: 0.80
Epoch 10/100
63/63 [=====] - 66s 1s/step - loss: 1.1784 - accuracy: 0.79
Epoch 11/100
63/63 [=====] - 65s 1s/step - loss: 1.1738 - accuracy: 0.81
Epoch 12/100
63/63 [=====] - 65s 1s/step - loss: 1.1619 - accuracy: 0.82
Epoch 13/100
63/63 [=====] - 65s 1s/step - loss: 1.1772 - accuracy: 0.80
Epoch 14/100
63/63 [=====] - 65s 1s/step - loss: 1.1622 - accuracy: 0.82
Epoch 15/100
63/63 [=====] - 65s 1s/step - loss: 1.1682 - accuracy: 0.81
Epoch 16/100
63/63 [=====] - 65s 1s/step - loss: 1.1812 - accuracy: 0.80
Epoch 17/100
63/63 [=====] - 66s 1s/step - loss: 1.1743 - accuracy: 0.81
Epoch 18/100
63/63 [=====] - 66s 1s/step - loss: 1.1845 - accuracy: 0.80
Epoch 19/100
63/63 [=====] - 66s 1s/step - loss: 1.1704 - accuracy: 0.82
Epoch 20/100
63/63 [=====] - 65s 1s/step - loss: 1.1688 - accuracy: 0.82
Epoch 21/100
63/63 [=====] - 65s 1s/step - loss: 1.1933 - accuracy: 0.79
Epoch 22/100
63/63 [=====] - 66s 1s/step - loss: 1.1926 - accuracy: 0.80
Epoch 23/100
63/63 [=====] - 66s 1s/step - loss: 1.1874 - accuracy: 0.80
Epoch 24/100
63/63 [=====] - 66s 1s/step - loss: 1.1837 - accuracy: 0.81
Epoch 25/100
63/63 [=====] - 65s 1s/step - loss: 1.1895 - accuracy: 0.80
Epoch 26/100
63/63 [=====] - 67s 1s/step - loss: 1.1878 - accuracy: 0.81
Epoch 27/100
63/63 [=====] - 69s 1s/step - loss: 1.1843 - accuracy: 0.81
Epoch 28/100
63/63 [=====] - 65s 1s/step - loss: 1.1890 - accuracy: 0.81
Epoch 29/100
63/63 [=====] - 65s 1s/step - loss: 1.1848 - accuracy: 0.81
Epoch 30/100
63/63 [=====] - 65s 1s/step - loss: 1.1869 - accuracy: 0.81
Epoch 31/100
```

```
63/63 [=====] - 68s 1s/step - loss: 1.1905 - accuracy: 0.81
Epoch 32/100
63/63 [=====] - 65s 1s/step - loss: 1.1942 - accuracy: 0.80
Epoch 33/100
63/63 [=====] - 69s 1s/step - loss: 1.1852 - accuracy: 0.82
Epoch 34/100
63/63 [=====] - 65s 1s/step - loss: 1.1882 - accuracy: 0.81
Epoch 35/100
63/63 [=====] - 67s 1s/step - loss: 1.1921 - accuracy: 0.81
Epoch 36/100
63/63 [=====] - 68s 1s/step - loss: 1.1831 - accuracy: 0.82
Epoch 37/100
63/63 [=====] - 65s 1s/step - loss: 1.1826 - accuracy: 0.82
Epoch 38/100
63/63 [=====] - 65s 1s/step - loss: 1.1933 - accuracy: 0.81
Epoch 39/100
63/63 [=====] - 65s 1s/step - loss: 1.1894 - accuracy: 0.81
Epoch 40/100
63/63 [=====] - 68s 1s/step - loss: 1.1974 - accuracy: 0.81
Epoch 41/100
63/63 [=====] - 65s 1s/step - loss: 1.1939 - accuracy: 0.81
Epoch 42/100
63/63 [=====] - 65s 1s/step - loss: 1.2023 - accuracy: 0.80
Epoch 43/100
63/63 [=====] - 69s 1s/step - loss: 1.1973 - accuracy: 0.81
Epoch 44/100
63/63 [=====] - 68s 1s/step - loss: 1.1982 - accuracy: 0.81
Epoch 45/100
63/63 [=====] - 65s 1s/step - loss: 1.1901 - accuracy: 0.82
Epoch 46/100
63/63 [=====] - 68s 1s/step - loss: 1.1947 - accuracy: 0.81
Epoch 47/100
63/63 [=====] - 66s 1s/step - loss: 1.1903 - accuracy: 0.82
Epoch 48/100
63/63 [=====] - 68s 1s/step - loss: 1.1912 - accuracy: 0.82
Epoch 49/100
63/63 [=====] - 69s 1s/step - loss: 1.1982 - accuracy: 0.81
Epoch 50/100
63/63 [=====] - 68s 1s/step - loss: 1.1924 - accuracy: 0.82
Epoch 51/100
63/63 [=====] - 65s 1s/step - loss: 1.1993 - accuracy: 0.81
Epoch 52/100
63/63 [=====] - 69s 1s/step - loss: 1.1961 - accuracy: 0.81
Epoch 53/100
63/63 [=====] - 67s 1s/step - loss: 1.1892 - accuracy: 0.82
Epoch 54/100
63/63 [=====] - 70s 1s/step - loss: 1.1952 - accuracy: 0.81
Epoch 55/100
63/63 [=====] - 69s 1s/step - loss: 1.1907 - accuracy: 0.82
Epoch 56/100
63/63 [=====] - 68s 1s/step - loss: 1.1849 - accuracy: 0.83
Epoch 57/100
63/63 [=====] - 66s 1s/step - loss: 1.2156 - accuracy: 0.79
Epoch 58/100
63/63 [=====] - 66s 1s/step - loss: 1.1819 - accuracy: 0.83
Epoch 59/100
63/63 [=====] - 66s 1s/step - loss: 1.2009 - accuracy: 0.81
Epoch 60/100
63/63 [=====] - 66s 1s/step - loss: 1.1964 - accuracy: 0.81
Epoch 61/100
63/63 [=====] - 66s 1s/step - loss: 1.2048 - accuracy: 0.80
Epoch 62/100
```

```
[...]
63/63 [=====] - 65s 1s/step - loss: 1.2020 - accuracy: 0.81
Epoch 63/100
63/63 [=====] - 66s 1s/step - loss: 1.2083 - accuracy: 0.80
Epoch 64/100
63/63 [=====] - 66s 1s/step - loss: 1.1910 - accuracy: 0.82
Epoch 65/100
63/63 [=====] - 66s 1s/step - loss: 1.2055 - accuracy: 0.80
Epoch 66/100
63/63 [=====] - 65s 1s/step - loss: 1.1991 - accuracy: 0.81
Epoch 67/100
63/63 [=====] - 65s 1s/step - loss: 1.2024 - accuracy: 0.81
Epoch 68/100
63/63 [=====] - 65s 1s/step - loss: 1.2027 - accuracy: 0.81
Epoch 69/100
63/63 [=====] - 65s 1s/step - loss: 1.1942 - accuracy: 0.82
Epoch 70/100
63/63 [=====] - 65s 1s/step - loss: 1.1967 - accuracy: 0.81
Epoch 71/100
63/63 [=====] - 65s 1s/step - loss: 1.1951 - accuracy: 0.82
Epoch 72/100
63/63 [=====] - 65s 1s/step - loss: 1.2096 - accuracy: 0.80
Epoch 73/100
```

```
model4.save('/content/drive/My Drive/SMARTBIN DVM/modelos/modelopb_4')
```

```
↳ INFO:tensorflow:Assets written to: /content/drive/My Drive/SMARTBIN DVM/modelos/mode
INFO:tensorflow:Assets written to: /content/drive/My Drive/SMARTBIN DVM/modelos/mode
Epochn /6/100
```

```
#Visualización de la precisión del entrenamiento y de la validacion
```

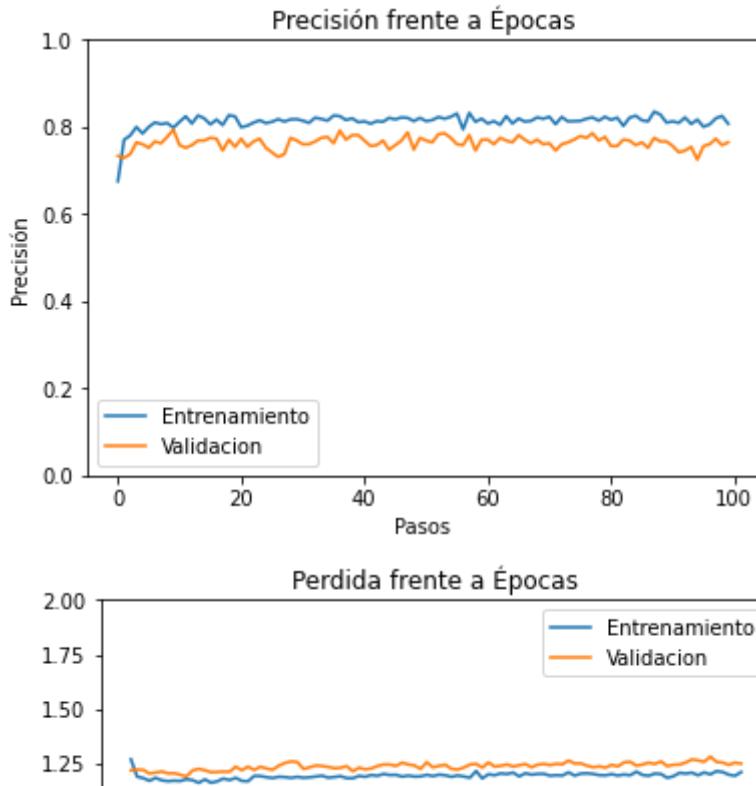
```
plt.figure()
plt.ylabel("Precisión ")
plt.xlabel("Pasos")
plt.ylim([0,1])
plt.plot(hist4["accuracy"])
plt.plot(hist4["val_accuracy"])
plt.legend(['Entrenamiento','Validacion']);
plt.title ("Precisión frente a Épocas")
```

```
#Valor de la función de perdida de la etapa de entrenamiento y validacion
```

```
plt.figure()
plt.ylabel("Perdida ")
plt.xlabel("Pasos ")
plt.ylim([0,2])
plt.plot(hist4["loss"])
plt.plot(hist4["val_loss"],)
plt.legend(['Entrenamiento','Validacion']);
plt.title ("Perdida frente a Épocas")
```

```
↳
```

Text(0.5, 1.0, 'Perdida frente a Épocas')



Comparativa de la precisión y perdida entre estos 4 experimentos.

```
#Visualización de la precisión del entrenamiento de estas 4 pruebas superpuestas
plt.figure()
plt.ylabel("Precisión ")
plt.xlabel("Epocas")
plt.ylim([0,1])
plt.plot(hist12["accuracy"])
plt.plot(hist2["accuracy"])
plt.plot(hist3["accuracy"])
plt.plot(hist4["accuracy"])
plt.legend(['ADAM+SGD', 'ADAM', 'SGD', 'NADAM']);
plt.title ("Precisión de entrenamiento frente a Épocas (lr=0.01)")
```

```
#Visualización de la precisión de validacion de estas 4 pruebas superpuestas
plt.figure()
plt.ylabel("Precisión ")
plt.xlabel("Epocas")
plt.ylim([0,1])
plt.plot(hist12["val_accuracy"])
plt.plot(hist2["val_accuracy"])
plt.plot(hist3["val_accuracy"])
plt.plot(hist4["val_accuracy"])
plt.legend(['ADAM+SGD', 'ADAM', 'SGD', 'NADAM']);
plt.title ("Precisión de validación frente a Épocas (lr=0.01)")
```

```
#Valor de la funcion de perdida de la etapa de entrenamiento de estas 4 pruebas superpues
plt.figure()
```

```
plt.ylabel("Perdida ")
plt.xlabel("Epocas ")
plt.ylim([0,2])
plt.plot(hist12["loss"])
plt.plot(hist2["loss"])
plt.plot(hist3["loss"])
plt.plot(hist4["loss"])
plt.legend(['ADAM+SGD','ADAM','SGD','NADAM']);
plt.title ("Perdida de entrenamiento frente a Épocas (lr=0.01)")
```

```
#Valor de la funcion de perdida de la etapa de validación de estas 4 pruebas superpuestas
```

```
plt.figure()
plt.ylabel("Perdida ")
plt.xlabel("Epocas ")
plt.ylim([0,2])
plt.plot(hist12["val_loss"])
plt.plot(hist2["val_loss"])
plt.plot(hist3["val_loss"])
plt.plot(hist4["val_loss"])
plt.legend(['ADAM+SGD','ADAM','SGD','NADAM']);
plt.title ("Perdida de validación frente a Épocas (lr=0.01)")
```

```
print ("Maximos de precisión durante el entrenamiento alcanzados por:")
print("ADAM+SGD: " + str(round(max(hist12["accuracy"]),4)*100)+"%")
print("ADAM: " + str(round(max(hist2["accuracy"]),4)*100)+"%")
print("SGD: " + str(round(max(hist3["accuracy"]),4)*100)+"%")
print("NADAM: " + str(np.round_(100*max(hist4["accuracy"])),2))+"%"
```

```
print ("Maximos de precisión durante la validación alcanzados por:")
print("ADAM+SGD: " + str(round(max(hist12["val_accuracy"]),4)*100)+"%")
print("ADAM: " + str(np.round(100*max(hist2["val_accuracy"])),2))+"%")
print("SGD: " + str(round(max(hist3["val_accuracy"]),4)*100)+"%")
print("NADAM: " + str(np.round(100*max(hist4["val_accuracy"])),2))+"%"
```

⇨

Como se comprueba en las gráficas anteriores, el optimizador de tipo SGD es el que ha obtenido unos mejores resultados, obteniendo una precisión máxima del 86.8% durante el entrenamiento y del 80.6% durante la validación. Por tanto se realizará un último experimento con este optimizador, reduciéndole la tasa de aprendizaje a 0.001 para comprobar si mejoran los resultados obtenidos.

## ▼ 5ºSGD (LR=0.001)

```
model5 = tf.keras.Sequential([
    # se construye una red neuronal con las siguientes capas
    tf.keras.layers.InputLayer(input_shape=IMAGE_SIZE + (3,)), # se crea una capa de entrada
    hub.KerasLayer(MODULE_HANDLE, trainable=False), # se añade INCEPTION V3 y se establece
    tf.keras.layers.Dropout(rate=0.5), # se añade una capa de Dropout con una desactivación
    tf.keras.layers.Dense(train_generator.num_classes, 'softmax'), # finalmente se añade una capa de salida
])
```

[https://colab.research.google.com/drive/18c2vxS\\_fOc8s3lZySKxOUZJeBs9mOrgq#scrollTo=5WtO6Ww80GOU&printMode=true](https://colab.research.google.com/drive/18c2vxS_fOc8s3lZySKxOUZJeBs9mOrgq#scrollTo=5WtO6Ww80GOU&printMode=true)

**15.2.2. *Código del script para generar un dataset***

```
1 import cv2
2 import threading
3 import time
4 def gstreamer_pipeline(
5     capture_width=1280,
6     capture_height=720,
7     display_width=1280,
8     display_height=720,
9     framerate=30,
10    flip_method=0,
11 ):
12     return (
13         "nvarguscamerasrc ! "
14         "video/x-raw(memory:NVMM), "
15         "width=(int)%d, height=(int)%d, "
16         "format=(string)NV12, framerate=(fraction)%d/1 ! "
17         "nvvidconv flip-method=%d ! "
18         "video/x-raw, width=(int)%d, height=(int)%d, format=(string)BGRx ! "
19         "videoconvert ! "
20         "video/x-raw, format=(string)BGR ! appsink"
21     % (
22         capture_width,
23         capture_height,
24         framerate,
25         flip_method,
26         display_width,
27         display_height,
28     )
29 )
30
31
32 cap = cv2.VideoCapture(gstreamer_pipeline(flip_method=0), cv2.CAP_GSTREAMER)
33
34 def mostrar():
35     # To flip the image, modify the flip_method parameter (0 and 2 are the most
36     # common)
37     print(gstreamer_pipeline(flip_method=0))
38
39     time.sleep(2)
40     if cap.isOpened():
41         window_handle = cv2.namedWindow("Smartbin", cv2.WINDOW_AUTOSIZE)
42         # Window
43         while cv2.getWindowProperty("Smartbin", 0) >= 0:
44             e.wait()
45             e.clear()
46             ret2, img2 = cap.read()
47             cv2.imshow("Smartbin", img2)
48             e.set()
49             # This also acts as
50             keyCode = cv2.waitKey(30) & 0xFF
51             # Stop the program on the ESC key
52
53             if keyCode == 27:
54                 break
55             cap.release()
56             cv2.destroyAllWindows()
57             exit()
58     else:
59         print("Unable to open camera")
```

```
60
61 def capturar(ruta):
62     ret, img = cap.read()
63     cv2.imwrite(ruta, img)
64
65
66
67
68
69 if __name__ == "__main__":
70     numplastico=38
71     numvidrio=6
72     nummetal=0
73     numpapel=8
74     numcarton=26
75     #threading.Thread(target=mostrar,name='Hilo-mostrar_camara')
76     entrada ='S'
77     e=threading.Event()
78     hilo = threading.Thread(target=mostrar,name='Hilo-mostrar_camara')
79     hilo.start()
80     e.set()
81     while (not (entrada=='F')):
82
83         entrada=input("Introduce el tipo de residuo Plastico->P || Papel->L || Vidrio->V
|| Metal->M || Carton->C || Finalizar->F      ")
84         if entrada=='P':
85             ruta= '/media/tfgdvm/pendrive/Smartbin-IA/dataset propio/plastico/plastico'
86             +str(numplastico) + '.jpg'
87             e.wait()
88             e.clear()
89             print('capturando')
90             time.sleep(1)
91             capturar(ruta)
92             e.set()
93             numplastico+=1
94             if entrada=='L':
95                 ruta= '/media/tfgdvm/pendrive/Smartbin-IA/dataset
96                 propio/papel/papel'+str(numpapel)+'.jpg'
97                 e.wait()
98                 e.clear()
99                 print('capturando')
100                time.sleep(1)
101                capturar(ruta)
102                e.set()
103                numpapel+=1
104                if entrada=='C':
105                    ruta= '/media/tfgdvm/pendrive/Smartbin-IA/dataset
106                    propio/carton/carton'+str(numcarton)+'.jpg'
107                    e.wait()
108                    e.clear()
109                    print('capturando')
110                    time.sleep(1)
111                    capturar(ruta)
112                    e.set()
113                    numcarton+=1
114                    if entrada=='M':
115                        ruta= '/media/tfgdvm/pendrive/Smartbin-IA/dataset
116                        propio/metal/metal'+str(nummetal) + '.jpg'
117                        e.wait()
118                        e.clear()
```

```
115     print('capturando')
116     time.sleep(1)
117     capturar(ruta)
118     e.set()
119     nummetal+=1
120     if entrada=='V':
121         ruta= '/media/tfgdvm/pendrive/Smartbin-IA/dataset propio/vidrio/vidrio'
122         +str(numvidrio) + '.jpg'
123         e.wait()
124         e.clear()
125         print('capturando')
126         time.sleep(1)
127         capturar(ruta)
128         e.set()
129         numvidrio+=1
130     if entrada=='F':
131         cap.release()
132         cv2.destroyAllWindows()
133         hilo.join()
134         exit()
135
136
```

**15.2.3.     *Código del Jetson Nano***

```
1
2 import camara #script para gestion de la camara
3 import clasificador #script para realizar la inferencia sobre la red neuronal
4 import time #libreria para manejar tiempo
5 import tensorflow as tf #libreria de IA
6 import comunicacion as com #script para gestion de comunicaciones
7 import serial #libreria para comunicaciones seria
8 import shutil #libreria para gestion de archivos en linux
9
10
11 num_procesados=[0,0,0,0,0,0]
12
13 modelo_ruta='/media/tfgdvm/pendrive/Smartbin-IA/modelos/modelo_lite/modelolite_1'
#ruta donde se encuentra el modelo (la imagen de la red neuronal) a importar
14 imagen_ruta ='/media/tfgdvm/pendrive/Smartbin-IA/residuos procesados/' #ruta donde
guardar las imagenes de los residuos procesados
15
16
17 interpreter = tf.lite.Interpreter(modelo_ruta) #se carga el modelo de la red
18 interpreter.allocate_tensors()
19
20 #si se establecen las conexiones serie tanto con el Arduino como con el ESP32
comienza el programa
21 with serial.Serial('/dev/serial/by-id/usb-1a86_USB2.0-Serial-if00-port0', 9600,
timeout=1) as mega:
22     with serial.Serial('/dev/serial/by-id/usb-
Silicon_Labs_CP2102_USB_to_UART_Bridge_Controller_0001-if00-port0', 9600,timeout=1)
as esp32:
23
24     com.enviar('H',mega) # hace el homing en la papelera
25
26     while True: #bucle que se ejecutará constantemente mientras la papelera permanezca
encendida
27
28     camara.detectar_mov() #detecta movimiento
29
30
31     print ("Movimiento detectado")
32
33     time.sleep(2) # espera 2 segundos a que el residuo se asienta para evitar
posibles rebotes
34
35     ruta = imagen_ruta + 'residuo_actual.jpg'
36
37     print('capturando')
38     camara.capturar (ruta) #guarda la imagen en el directorio ruta provisionalmente
39
40
41     t = time.process_time() #guarda el tiempo actual
42     material=clasificador.clasificar(interpreter,ruta) #clasifica la imagen en ruta
mediante la red neuronal
43     print("Tiempo de clasificacion:" , time.process_time() - t) #indica el tiempo de
procesamiento de la red
44
45     vector_aux=['P','C','M','V','O','L'] #vector auxiliar de iniciales de cada
residuo (la 'L' corresponde al papel)
46
47     material_int= vector_aux.index(material[0]) # transforma el material de un
string a un int del 0 al 5
```

```
48     #(Plastico(P)-> 0 ,Carton (C)->1 ,Metal (M)->2, Vidrio (V)-> 3, Organico (O)->
49     4, Papel (L)-> 5)
50
50     ruta2= imagen_ruta + material+ '/' + material + str(num_procesados[material_int])
51     + '.jpg' # se establece la ruta del residuo en cuestión
51     shutil.move (ruta,ruta2) #se mueve la imagen del residuo procesado a su
51     directorio corrrespondiente para generar asi una base de imagenes de residuos
51     procesados
52     num_procesados[material_int]+=1 #se incrementa el numero de residuos procesados
52     de esa clase
53
54     print("Procesando" , material)
55
56     contenedor=material #el nombre del contenedor es igual al material salvo
57     if material=='Papel':#salvo en el caso del papel que va al contenedor del carton
57         contenedor='Carton'
58
59
60
61     print("Girando selector a posicion del contenedor de " ,contenedor)
62     com.enviar(contenedor[0],mega) #s envia al arduino mega la inicial del contenedor
62     al que debe de enviar el residuo
63     print("Abriendo compuertas")
64     com.enviar('B',mega) #abre compuertas
65     time.sleep(5) #espera 5 segundos a que caiga el residuo
66     print("Cerrando compuertas")
67
68     comandou = 'U' + str(vector_aux.index(contenedor[0])) # envía al ESP32 una 'U'
68     seguido de un numero del 0 al 4 indicando el tipo de residuo
69     print(comandou)
70
71     com.enviar(comandou, esp32) #envia el comando anterior para ejecutar la medicion
71     de ultrasonidos y el posterior envio a traves de LoRa
72     com.enviar('S',mega) #cierra compuertas
73
74
75
76
77
78
79
```

```
1 import serial
2
3 def enviar(comando,ser): #función enviar , recibe el caracter a enviar (comando) y el
4     # objeto serial (ser) que puede hacer referencia tanto al mega como al esp32
5     ser.write(bytes(comando,'utf-8'))#escribe en el serial el comando
6     print('enviado',comando)
7     while (not ser.read().decode('utf-8')=='F'): #mientras no reciba una 'F' no sale de
8         #este bucle
9         print('Ejecutando accion')
10        print('F')
```

```
1 import cv2 #librería para controlar la camara
2 import time #libreria para gestionar el tiempo
3
4
5
6
7
8 def gstreamer_pipeline( #características de captura y visualizacion para detectar
9     el movimiento
10    capture_width=320, #baja resolución para reducir la carga de calculo (recorre
11    pixel a pixel comprobando si ha cambiado)
12    capture_height=240,
13    display_width=320,
14    display_height=240,
15    framerate=15, #baja tasa de fotogramas por segundo, no es un factor
determinante solamente para visualizar la camara
16    flip_method=0,
17 ):
18    return (
19        "nvarguscamerasrc ! "
20        "video/x-raw(memory:NVMM), "
21        "width=(int)%d, height=(int)%d, "
22        "format=(string)NV12, framerate=(fraction)%d/1 ! "
23        "nvvidconv flip-method=%d ! "
24        "video/x-raw, width=(int)%d, height=(int)%d, format=(string)BGRx ! "
25        "videoconvert ! "
26        "video/x-raw, format=(string)BGR ! appsink"
27        % (
28            capture_width,
29            capture_height,
30            framerate,
31            flip_method,
32            display_width,
33            display_height,
34        )
35 def gstreamer_pipeline_captura( #características para la captura de la imagen del
residuo
36    capture_width=1920, #resolución alta pues será sobre esta imagen sobre la que se
realice la clasificacion
37    capture_height=1080,
38    display_width=1920,
39    display_height=1080,
40    framerate=30, #no determinante
41    flip_method=0,
42 ):
43    return (
44        "nvarguscamerasrc ! "
45        "video/x-raw(memory:NVMM), "
46        "width=(int)%d, height=(int)%d, "
47        "format=(string)NV12, framerate=(fraction)%d/1 ! "
48        "nvvidconv flip-method=%d ! "
49        "video/x-raw, width=(int)%d, height=(int)%d, format=(string)BGRx ! "
50        "videoconvert ! "
51        "video/x-raw, format=(string)BGR ! appsink"
52        % (
53            capture_width,
54            capture_height,
55            framerate,
```

```
56         flip_method,
57         display_width,
58         display_height,
59     )
60 )
61 cap = cv2.VideoCapture(gststreamer_pipeline(flip_method=0), cv2.CAP_GSTREAMER) # se
ejecuta el metodo VideoCapture para iniciar la camara
62
63 def detectar_mov(): #metodo para detectar movimiento
64
65     sensibilidad=30000 #cantidad de pixeles que deben de cambiar para que se
    considere que ha habido movimiento
66
67     if cap.isOpened():# si esta activa la camara
68
69         window_handle = cv2.namedWindow("Smartbin", cv2.WINDOW_AUTOSIZE)#genera una
    ventana para visualizar la camara
70
71         ret1, img1 = cap.read() #lee una imagen y la guarda en el vector img1
72         mov=False
73         # Window
74         T0= time.time() #tiempo acutal
75
76         while cv2.getWindowProperty("Smartbin", 0) >= 0:
77             diferencia = 0 #variable donde guardar la cantidad de pixeles diferentes
78
79             ret2, img2 = cap.read() #guarda otro fotograma
80             cv2.imshow("Smartbin", img2) #lo muestra
81
82             frameDelta = cv2.absdiff(img1, img2) #calcula la diferencia entre dos
    vectores en este caso los pixeles que han cambiado entre la imagen inicial y la
    actual
83             #y guarda el vector diferencia en frameDelta
84             thresh = cv2.threshold(frameDelta, 25, 255, cv2.THRESH_BINARY)[1]
    #binariza la diferencia (frameDelta) para poder sumar los pixeles mas facilmente
85
86             if(thresh.sum())>sensibilidad and (time.time()-T0)>2: #suma los pixeles
    que han cambiado
87                 # y si es mayor que la sensibilidad y pasaron mas de 2s (tiempo para
    que la camara se inicie correctamente) ha detectado un movimiento
88
89                 print("Movimiento detectado!")
90                 break
91
92             else:
93                 print("No hay movimiento")
94             keyCode = cv2.waitKey(30) & 0xFF #si se pulsase la tecla ESC
95             if keyCode == 27:
96                 break #se sale del bucle
97             img1=img2 #si no se ha detectado movimiento la imagen de referencia es
    la anterior
98
99             cap.release() #se deja de usar la camara
100            cv2.destroyAllWindows() #se destruyen todas las ventanas
101        else:
102            print("No se pudo abrir la camara") #si no es capaz de abrir la camara
103        return mov
104
105
106
```

```
107| def capturar(ruta): #metodo para caputrar imagen
108|     cap = cv2.VideoCapture(gstreamer_pipeline_captura(flip_method=0),
109|     cv2.CAP_GSTREAMER) #inicia la camara con los parametros de captura
110|     ret, img = cap.read() #lee la imagen
111|     cv2.imwrite(ruta, img) #guarda la imagen en la ruta proporcionada
112|
113|
114|
115|
116|
117|
118|
119|
120|
```

```
1 from __future__ import absolute_import
2 from __future__ import division
3 from __future__ import print_function
4
5 import argparse
6 import numpy as np
7 from PIL import Image
8
9
10 def load_labels(filename): # función para cargar las etiquetas o clases en formato de texto
11     with open(filename, 'r') as f:
12         return [line.strip() for line in f.readlines()]
13
14
15
16 def clasificar(interpreter,imagen): #función a la que se le pasa la red (interpreter) y clasifica la imagen contenida en la ruta (imagen)
17
18
19     input_details = interpreter.get_input_details() #guarda las características de la entrada de la red
20     output_details = interpreter.get_output_details() #guarda las características de salida de la red
21
22     # NxHxWxC, H:1, W:2
23     height = input_details[0]['shape'][1] #altura que debe de tener la imagen para introducirla en la red (299)
24     width = input_details[0]['shape'][2] #ancho que debe de tener la imagen para introducirla en la red (299)
25
26
27     floating_model = input_details[0]['dtype'] == np.float32 #comprueba el que el tensor de entrada sea de tipo float 32
28
29
30     label_file='/media/tfgdvm/pendrive/Smartbin-IA/output_labels.txt' #ruta donde se encuentran las etiquetas de cada clase (permite transformar la salida de la red en una clase de forma textual)
31
32
33     input_mean=127.5
34
35     input_std=127.5
36
37
38
39     img = Image.open(imagen).resize((width, height)) # se abre la imagen alojada en la ruta proporcionada y se transforma su tamaño para adaptarlo al que acepta la red
40
41
42     input_data = np.expand_dims(img, axis=0) #se añaden tantas dimensiones como tiene la imagen
43
44     if floating_model:
45         input_data = (np.float32(input_data) - input_mean) / input_std
46
47     interpreter.set_tensor(input_details[0]['index'], input_data)
48
49     interpreter.invoke() #invoca al interprete del formato tf.lite
```

```

50
51     output_data = interpreter.get_tensor(output_details[0]['index']) #extrae en forma
52     de vector los valores que proporciona el tensor de la capa de salida
53     results = np.squeeze(output_data) #elimina del vector output_data todos los subset
54     de 1 dimension y
55     #por tanto quedaría un vector de 5 posiciones con los porcentajes (del 0 al 1) de
56     #que pertenezca a cada una de las 5 clases de residuos
57
58     top_k = results.argsort()[-5:][::-1] #en este caso son 5 clases
59     labels = load_labels(label_file) #se cargan las etiquetas
60
61     for i in top_k: #para cada una de las clases se visualiza el porcentaje de
62     #pertenercer a cada una de ellas
63         if floating_model:
64             print('{:08.6f}: {}'.format(float(results[i]), labels[i]))
65         else:
66             print('{:08.6f}: {}'.format(float(results[i] / 255.0), labels[i]))
67
68     if max(results)<0.6: #si el porcentaje más alto de pertenecer a una clase no supera
69     #el 60% se considera que el residuo es orgánico
70         material= 'Organico'
71         print('No se reconoce este residuo')
72     else: #de lo contrario
73         material = labels[np.argmax(results, axis=-1)] #el residuo es de la clase que la
74         red ha determinado que es
75
76         print("Es ",material)
77
78     return material #devuelve la clase a la que pertenece le residuo que se está
79     procesando
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94

```

**15.2.4.     *Código de Arduino Mega***

```
1 #include <Arduino.h>
2
3 //Pines del motor del selector NEMA17
4 int pasos = 12;                                //definimos como «entero» pin
5 int direccion = 11;                            //definimos como «entero» pin
6 int reset = 10;
7
8 //Pines de los motores de las compuertas 28BY J-48
9 int Motores[2][4]={{22,24,26,28} , // Fila 1--> Motor de la puerta 1 (4 pines) ;
10           {5,4,3,2} }; // Fila 2--> Motor de la puerta 2 (4 pines) ;
11
12
13 #define vel_sel 40000
14 #define vel_comp 600
15
16 int FC[3] ={52,50,48} ; //finales de carrera 0--> SELECTOR 1-->PUERTA 1 2-->
17 PUERTA 2
18
19 #define L_AZUL 36
20 #define L_AMAR 40
21 #define L_ROJO 42
22 #define L_VERDE 38
23 #define L_BLANCO 34
24
25 int posic_actual[3];//VECTOR DE POSICIONES ACTUALES 0->SELECTOR      1->PUERTA 1
26   2->PUERTA 2
27 int posic_plastico=560;
28 int posic_vidrio=420;
29 int posic_carton=280;
30 int posic_metal=140;
31 int posic_organico=0;    //=700
32
33 int Steps[3];
34 char entrada;
35
36 boolean Direction[3] ;
37
38 int NormalPaso [4][4] =
39 {
40   {1, 1, 0, 0},
41   {0, 1, 1, 0},
42   {0, 0, 1, 1},
43   {1, 0, 0, 1}
44 };
45
46 int Completopaso [4][4] =
47 {
48   {1, 0, 0, 0},
49   {0, 1, 0, 0},
50   {0, 0, 1, 0},
51   {0, 0, 0, 1}
52 };
53
54 int MedioPaso [ 8 ][ 4 ] =
55 {
56   { 1, 0, 0, 0},
57   {1, 1, 0, 0},
58   {0, 1, 0, 0},
```

```
57     {0, 1, 1, 0},  
58     {0, 0, 1, 0},  
59     {0, 0, 1, 1},  
60     {0, 0, 0, 1},  
61     {1, 0, 0, 1}  
62 };  
63  
64 void setup() {  
65     // start serial port at 9600 bps:  
66     Serial.begin(9600);  
67  
68     pinMode(pasos, OUTPUT);          //definimos pasos como salida digital  
69     pinMode(direccion, OUTPUT);      //definimos direccion como salida digital  
70     pinMode(reset, OUTPUT);         //definimos reset como salida digital  
71  
72     for (int fil=0;fil<2;fil++){  
73         for (int col=0;col<5;col++){  
74             pinMode(Motores[fil][col], OUTPUT);  
75         }  
76     }  
77     pinMode(L_AZUL, OUTPUT);  
78     pinMode(L_AMAR, OUTPUT);  
79     pinMode(L_ROJO, OUTPUT);  
80     pinMode(L_VERDE, OUTPUT);  
81     pinMode(L_BLANCO, OUTPUT);  
82     pinMode(FC[0], INPUT);  
83     pinMode(FC[1], INPUT);  
84     pinMode(FC[2], INPUT);  
85     while (!Serial) {  
86         ; // wait for serial port to connect.  
87     }  
88 }  
89 void SetDirection(int x)  
90 {  
91     if(Direction[x])  
92         Steps[x]++;  
93     else {  
94         Steps[x]--;  
95     }  
96  
97     Steps[x] = ( Steps[x] + 4 ) % 4 ;  
98 }  
99  
100 void stepper(int fil)           //Avanza un paso  
101 {  
102  
103     digitalWrite(Motores[fil][0], NormalPaso[Steps[fil]][ 0 ] );  
104  
105     digitalWrite(Motores[fil][1], NormalPaso[Steps[fil]][ 1 ] );  
106  
107  
108     digitalWrite(Motores[fil][2], NormalPaso[Steps[fil]][ 2 ] );  
109  
110     digitalWrite(Motores[fil][3], NormalPaso[Steps[fil]][ 3 ] );  
111  
112  
113  
114  
115  
116     SetDirection(fil);
```

```
117
118
119
120
121
122     }
123
124 void avanza (int x){
125
126     Direction[x]=true;
127     posic_actual[x+1]++;
128     stepper(x) ;      // Avanza un paso
129     delayMicroseconds (vel_comp) ;
130
131
132 }
133
134 void retrocede (int x){
135
136     Direction[x]=false;
137     posic_actual[x+1]--;
138     stepper(x) ;      // retrocede un paso
139     delayMicroseconds (vel_comp) ;
140
141
142 }
143
144 void reset_nema(){
145     digitalWrite(reset, LOW);           //Mientras reset este en LOW, el motor
146     permanece apagado                //Retardo en la instruccion
147     delay(2000);                     //Cuando reset se encuentre en HIGH el motor
148     digitalWrite(reset, HIGH);        //arranca
149
150
151 void homingsel(){
152     reset_nema();
153     digitalWrite(direccion, HIGH);   //mandamos direccion al servo
154     while (  digitalRead(FC[0])==LOW ) //normalmente cerrado
155     {
156         digitalWrite(pasos, HIGH);    // ponemos a high «pasos»
157         digitalWrite(pasos, LOW);     // ponemos a low «pasos»
158         delayMicroseconds(vel_sel);  // leemos la referencia de velocidad
159     }
160     digitalWrite(reset, LOW);        //Mientras reset este en LOW, el motor
161     permanece apagado
162     posic_actual[0]=0;
163
164 }
165
166 void homing_comp(){
167     while (not ( digitalRead(FC[1])==HIGH && digitalRead(FC[2])==HIGH ) )
168     //normalmente cerrado
169     {
170         if (digitalRead(FC[1]) ==LOW) {
171             retrocede(0);
172         }
173         if (digitalRead(FC[2]) ==LOW) {
```

```

173         retrocede(1);
174     }
175 }
176 posic_actual[1,2]=0;
177 Serial.write('F');
178 delay(300);
179 }
180
181 void loop() {
182     digitalWrite(L_AMAR, LOW);
183     digitalWrite(L_ROJO, LOW);
184     digitalWrite(L_VERDE, LOW);
185     digitalWrite(L_AZUL, LOW);
186
187
188
189
190     if (Serial.available() > 0) {
191
192         int entrada = Serial.read();
193         if (entrada=='P'){
194             digitalWrite(L_AMAR, HIGH);
195             reset_nema();
196             while(not (posic_actual[0]==posic_plastico))
197             {
198                 //Serial.write(posic_actual);
199                 if (posic_actual[0]<posic_plastico){
200                     digitalWrite(direccion, LOW); //mandamos direccion al servo
201                     posic_actual[0]++;
202
203                 }
204                 else{
205                     digitalWrite(direccion, HIGH); //mandamos direccion al servo
206                     posic_actual[0]--;
207                 }
208                 digitalWrite(pasos, HIGH); // ponemos a high «pasos»
209                 digitalWrite(pasos, LOW); // ponemos a low «pasos»
210                 delayMicroseconds(vel_sel); // leemos la referencia de velocidad
211
212             }
213             digitalWrite(reset, LOW); //Mientras reset este en LOW, el
motor permanece apagado
214             //Serial.write(posic_actual);
215             Serial.write('F');
216             delay(300);
217
218     }
219     if (entrada=='C'){
220         digitalWrite(L_AZUL, HIGH);
221         reset_nema();
222         while(not (posic_actual[0]==posic_carton))
223         {
224             if (posic_actual[0]<posic_carton){
225                 digitalWrite(direccion, LOW); //mandamos direccion al servo
226                 posic_actual[0]++;
227
228             }
229             else{
230                 digitalWrite(direccion, HIGH); //mandamos direccion al servo
231                 posic_actual[0]--;

```

```
232         }
233         digitalWrite(pasos, HIGH);           // ponemos a high «pasos»
234         digitalWrite(pasos, LOW);            // ponemos a low «pasos»
235         delayMicroseconds(vel_sel); // leemos la referencia de velocidad
236
237     }
238
239
240     digitalWrite(reset, LOW);           //Mientras reset este en LOW, el
motor permanece apagado
241     Serial.write('F');
242     delay(300);
243
244 }
245 if (entrada=='M'){
246     digitalWrite(L_ROJO, HIGH);
247     reset_nema();
248     while(not (posic_actual[0]==posic_metal))
249     {
250         if (posic_actual[0]<posic_metal){
251             digitalWrite(direccion, LOW); //mandamos direccion al servo
252             posic_actual[0]++;
253
254         }
255         else{
256             digitalWrite(direccion, HIGH); //mandamos direccion al servo
257             posic_actual[0]--;
258         }
259         digitalWrite(pasos, HIGH);           // ponemos a high «pasos»
260         digitalWrite(pasos, LOW);            // ponemos a low «pasos»
261         delayMicroseconds(vel_sel); // leemos la referencia de velocidad
262
263     }
264     digitalWrite(reset, LOW);           //Mientras reset este en LOW, el
motor permanece apagado
265     Serial.write('F');
266     delay(300);
267
268 }
269 if (entrada=='V'){
270     digitalWrite(L_VERDE, HIGH);
271     reset_nema();
272     while(not (posic_actual[0]==posic_vidrio))
273     {
274         if (posic_actual[0]<posic_vidrio){
275             digitalWrite(direccion, LOW); //mandamos direccion al servo
276             posic_actual[0]++;
277
278         }
279         else{
280             digitalWrite(direccion, HIGH); //mandamos direccion al servo
281             posic_actual[0]--;
282         }
283         digitalWrite(pasos, HIGH);           // ponemos a high «pasos»
284         digitalWrite(pasos, LOW);            // ponemos a low «pasos»
285         delayMicroseconds(vel_sel); // leemos la referencia de velocidad
286
287     }
288     digitalWrite(reset, LOW);           //Mientras reset este en LOW, el
motor permanece apagado
```

```

289         Serial.write('F');
290         delay(300);
291     }
292
293     if (entrada=='O'){
294         digitalWrite(L_BLANCO, HIGH);
295         reset_nema();
296         while(not (posic_actual[0]==posic_organico))
297         {
298             if (posic_actual[0]<posic_organico){
299                 digitalWrite(direccion, LOW); //mandamos direccion al servo
300                 posic_actual[0]++;
301
302             }
303             else{
304                 digitalWrite(direccion, HIGH); //mandamos direccion al servo
305                 posic_actual[0]--;
306             }
307             digitalWrite(pasos, HIGH); // ponemos a high «pasos»
308             digitalWrite(pasos, LOW); // ponemos a low «pasos»
309             delayMicroseconds(vel_sel); // leemos la referencia de velocidad
310
311         }
312         digitalWrite(reset, LOW); //Mientras reset este en LOW, el
motor permanece apagado
313         Serial.write('F');
314         delay(300);
315     }
316
317 }
318
319
320     if (entrada=='H'){ //hacer homing
321
322         digitalWrite(L_AMAR, HIGH);
323         digitalWrite(L_ROJO, HIGH);
324         digitalWrite(L_VERDE, HIGH);
325         digitalWrite(L_AZUL, HIGH);
326         homingsel();
327         homing_comp();
328         Serial.write('F');
329         delay(300);
330
331     }
332     if (entrada=='B'){
333         digitalWrite(L_AMAR, HIGH);
334         digitalWrite(L_AZUL, HIGH);
335
336         while (posic_actual[1]<3400 and posic_actual[2]<3400 ){
337             avanza(0);
338             avanza(1);
339         }
340         Serial.write('F');
341         delay(300);
342
343     }
344     if (entrada=='S'){ //subir compuertas
345         homing_comp();
346
347 }

```

```
348 }  
349 }  
350  
351  
352  
353  
354  
355  
356  
357  
358  
359  
360  
361  
362  
363  
364
```

**15.2.5.      *Código del emisor LoRa***

```
1 //Codigo basado en el proyecto de Rui Santos
2
3 *****
4   Rui Santos
5   Complete project details at https://RandomNerdTutorials.com/esp32-lora-sensor-web-server/
6
7   Permission is hereby granted, free of charge, to any person obtaining a copy
8   of this software and associated documentation files.
9
10  The above copyright notice and this permission notice shall be included in all
11  copies or substantial portions of the Software.
12 *****/
13
14
15 #include <Arduino.h>
16
17 //Librerias para la gestión del LoRa
18 #include <SPI.h>
19 #include <LoRa.h>
20
21 //Librerias para la visualizacion el la OLED
22 #include <Wire.h>
23 #include <Adafruit_GFX.h>
24 #include <Adafruit_SSD1306.h>
25
26
27 //se definen los pines para el modulo LoRa que incorpora el TTGO
28 #define SCK 5 //señal de reloj
29 #define MISO 19 //((Master Input Slave Output) señal de entrada
30 #define MOSI 27 //((Master Output Slave Input) señal de salida
31 #define SS 18 //((Slave Select) selecciona el esclavo al que se envían
32 #define RST 14 //boton de reset
33 #define DIO0 26 //
34
35 #define BAND 866E6 //se definen la banda de transmisión en europa es la 866E6
36
37 //Se definen los pines de la OLED
38 #define OLED_SDA 4
39 #define OLED_SCL 15
40 #define OLED_RST 16
41 #define SCREEN_WIDTH 128 // OLED ancho de la pantalla en pixeles
42 #define SCREEN_HEIGHT 64 // OLED alto de la pantalla en pixeles
43
44 int readingID = 0; //variable para el ID del mensaje
45
46 int counter = 0; //contador
47 String LoRaMessage = ""; //en esta string se irá creando el mensaje para enviar
48
49 // vector donde se guardarán los niveles de los 5 depositos
50 float nivel[5]={0,0,0,0,0} ; //0->Plastico ; 1->Papel/Carton ; 2->Metal ;3-> Vidrio
;4->Organico
51 int material ;
52 const int EchoPin = 13; // el pin ECHO del sensor de ultrasonidos se conecta al pin
13
53 const int TriggerPin = 12; //el trigger se conecta al pin 12
54
55 //se crea un objeto display de tipo Adafruit_SSD1306 con las características de la
pantalla
56 Adafruit_SSD1306 display(SCREEN_WIDTH, SCREEN_HEIGHT, &Wire, OLED_RST);
```

```
57
58
59 //Inicializa la pantalla OLED
60 void startOLED(){
61     //resetea el OLED
62     pinMode(OLED_RST, OUTPUT);
63     digitalWrite(OLED_RST, LOW);
64     delay(20);
65     digitalWrite(OLED_RST, HIGH);
66
67     //inicializa el OLED
68     Wire.begin(OLED_SDA, OLED_SCL);
69     if(!display.begin(SSD1306_SWITCHCAPVCC, 0x3c, false, false)) { // Address 0x3C for
128x32
70         Serial.println(F("SSD1306 allocation failed"));
71         for(;); // Don't proceed, loop forever
72     }
73     display.clearDisplay(); //limpia la pantalla
74     display.setTextColor(WHITE); //establece como blanco el color de escritura
75     display.setTextSize(1); //establece el tamaño del texto en una fila
76     display.setCursor(0,0); //pone el cursor en el origen (parte superior izquierda)
77     display.print("Emisor LoRa");
78 }
79
80 //Inicializa el modulo LoRa
81 void startLoRA(){
82     int counter;
83
84     SPI.begin(SCK, MISO, MOSI, SS); //establece los pines correspondientes
85     LoRa.setPins(SS, RST, DIO0); //establece el modulo LoRa
86
87 //hace una prueba para comprobar si se establece conexión en un tiempo determinado
88     while (!LoRa.begin(BAND) && counter < 10) { //envía hasta 10 paquetes mientras no
haya establecido conexión
89         Serial.print(".");
90         counter++;
91         delay(500);
92     }
93     if (counter == 10) { //si se han enviado los 10 paquetes quiere decir que se agoto
el tiempo de conexión
94         Serial.println("Error: fallo de inicializacion"); //salta un error
95     }
96     Serial.println("Inicializa LoRa OK!"); //si se establecio conexión en la pantalla
se muestra este mensaje
97     display.setCursor(0,10); //desplaza el cursor 10 columnas a la derecha
98     display.clearDisplay(); //limpia la pantalla
99     display.display(); //lo muestra
100    delay(2000);
101 }
102
103 //función que devuelve la distancia en cm proporcionada por el sensor de
ultrasonidos
104 int medir (int TriggerPin, int EchoPin) {
105     long duracion, distancia_Cm;
106
107     digitalWrite(TriggerPin, LOW); //para generar un pulso limpio pone a LOW 4us
108     delayMicroseconds(4);
109     digitalWrite(TriggerPin, HIGH); //genera Trigger (disparo) de 10us
110     delayMicroseconds(10);
111     digitalWrite(TriggerPin, LOW);
```

```
112
113     duracion = pulseIn(EchoPin, HIGH); //mide el tiempo entre pulsos, en
114     microsegundos
115     distancia_Cm = duracion * 10 / 292/ 2; //convierte a distancia, en cm
116     //5/292
117     return distancia_Cm;
118 }
119 // funcion que segun el material que se este procesando guarda el nivel medido por
120 // el ultrasonidos en su variable correspondiente
121 void getReadings(int material){
122     int cm = medir (TriggerPin, EchoPin); //obtiene la distancia en cm
123     Serial.print("Distancia: ");
124     Serial.println(cm);
125     float nivel_medido = (-2.56)*cm+ (112.82); //nivel maximo 100 % cuando la
126     distancia es 5 cm ; nivel minimo 0% cuando distancia es 44 cm
127     nivel[material]= nivel_medido;//el nivel medido se guarda en la variable del
128     nivel que se este clasificando
129     Serial.print("Nivel: ");
130     Serial.println(nivel_medido);
131 }
132 //función que crea el mensaje y lo envia mediante LoRa tambien visualiza la info en
133 //la pantalla
134 void sendReadings(int material) {
135     //se crea el mensaje intercalando los datos con marcadores
136     LoRaMessage = String(readingID) + "/" + String(material) + '&' +
137     String(nivel[material]);
138     // se envia el paquete
139     LoRa.beginPacket();
140     LoRa.print(LoRaMessage);
141     LoRa.endPacket();
142
143     display.clearDisplay(); //se limpia la OLED
144     display.setCursor(0,0); //se desplaza el cursor al origen
145     display.setTextSize(1); //se establece el tamaño de letra en 1 fila y 1 columna
146     display.print("Paquete " + String(readingID) + " enviado"); //se escribe
147     display.setCursor(0,10);
148     display.print("Plastico:"); //se escribe
149     display.setCursor(72,10); //se desplaza el cursor 72 pixeles a la derecha
150     display.print(nivel[0] ); //se escribe el nivel de plastico
151     display.setCursor(0,20); //se desplaza el cursor 10 pixeles mas abajo
152     display.print("Carton:");
153     display.setCursor(54,20);
154     display.print(nivel[1]);
155     display.setCursor(0,30);
156     display.print("Metal:");
157     display.setCursor(54,30);
158     display.print(nivel[2]);
159     display.setCursor(0,40);
160     display.print("Vidrio:");
161     display.setCursor(54,40);
162     display.print(nivel[3]);
163     display.setCursor(0,50);
164     display.print("Organico:");
165     display.setCursor(54,50);
166     display.print(nivel[4]);
```

```
166 display.display();
167 Serial.print("Sending packet: ");
168 Serial.println(readingID); //se muestra el ID tambien a traves del monitor serial
169 readingID++;
170 }
171
172 void setup() {
173
174     Serial.begin(9600); //inicializa la comunicacion serial a 9600 baudios
175     while (!Serial) {
176         ; // espera hasta que la comunicacion serial se establezca
177     }
178     pinMode(TriggerPin, OUTPUT); //se configura el pin del trigger como salida
179     pinMode(EchoPin, INPUT); //se configura el pin del echo como entrada
180
181     startOLED(); //se llama a la funcion para inicializar la OLED
182     startLoRA(); //se llama a la funcion para inicializar EL LORA
183 }
184
185 void loop() {
186
187     char buffer[2]; //se crea un buffer de caracteres de 2 posiciones
188
189     if (Serial.available() > 0) { //si el serial esta activo
190         int size = Serial.readBytesUntil('\n', buffer, 2); //lee los caracteres que
191         //entran por el serial hasta la siguiente linea (/n) y los guarda en el buffer (son 2
192         //bytes)
193         if (buffer[0] == 'U') { //si recibe una U ejecuta el proceso
194             material = int(buffer[1])-48; //convierte el segundo char recibido a int
195             getReadings(material); //lee el dato del ultrasonidos recibido,lo guarda en la
196             //variable material recibido
197             sendReadings(material); //envia los datos
198             Serial.write('F'); //envia por el serial una F para indicarle al arduino que ha
199             //finalizado
200             delay(300);
201         }
202     }
203 }
```

**15.2.6.     *Código del receptor LoRa***

```
1 //Codigo basado en el proyecto de Rui Santos
2
3 *****
4   Rui Santos
5   Complete project details at https://RandomNerdTutorials.com/esp32-lora-sensor-web-server/
6
7   Permission is hereby granted, free of charge, to any person obtaining a copy
8   of this software and associated documentation files.
9
10  The above copyright notice and this permission notice shall be included in all
11  copies or substantial portions of the Software.
12 *****/
13
14 #include <Arduino.h>
15 //Librerias para la conexión WIFI
16 #include <WiFi.h>
17 #include <ESPAsyncWebServer.h>
18
19 #include <SPIFFS.h>
20
21 //Librerias para la gestión del LoRa
22 #include <SPI.h>
23 #include <LoRa.h>
24
25 //Librerias para la visualización el la OLED
26 #include <Wire.h>
27 #include <Adafruit_GFX.h>
28 #include <Adafruit_SSD1306.h>
29
30 // Librerias para el servidor NTP
31 #include <NTPClient.h>
32 #include <WiFiUdp.h>
33
34 //definimos los pines para el modulo LoRa que incorpora el TTGO
35 #define SCK 5 //señal de reloj
36 #define MISO 19 // (Master Input Slave Output) señal de entrada
37 #define MOSI 27 // (Master Output Slave Input) señal de salida
38 #define SS 18 // (Slave Select) selecciona el esclavo al que se envían
39 #define RST 14 //boton de reset
40 #define DIO0 26 //
41
42
43 #define BAND 866E6 //definimos la banda de transmisión en europa es la 866E6
44
45 //Definimos los pines de la OLED
46 #define OLED_SDA 4
47 #define OLED_SCL 15
48 #define OLED_RST 16
49 #define SCREEN_WIDTH 128 // OLED ancho de la pantalla en pixeles
50 #define SCREEN_HEIGHT 64 // OLED alto de la pantalla en pixeles
51
52 //definimos los parametros para la conexión WIFI
53 const char* ssid      = "Yo";
54 const char* contrasena = "aprueboTFG";
55
56 // Definimos el cliente NTP para recibir la fecha y la hora
57 WiFiUDP ntpUDP;
58 NTPClient timeClient(ntpUDP);
59
```

```
60 // Variables para guardar el dia y la fecha
61 String formattedDate;
62 String dia;
63 String hora;
64 String timestamp;
65
66
67 // Initialize variables to get and save LoRa data
68 int rssi;
69 String loraMessage; //en esta variable se guarda el mensaje recibido
70
71
72 String nivel[5] ; //vector de string donde se guardaran y actualizaran los valores
    de los niveles de cada deposito
73 int material; //se guarda el numero del material (la posicion el vector nivel) que
    se esta clasificando 0->Plastico ; 1->Papel/Carton ; 2->Metal ;3-> Vidrio ;4-
    >Organico
74
75 String readingID; //ID recibido
76
77 // Se crea un servidor en el puerto 80
78 AsyncWebServer server(80);
79
80 //se crea un objeto display de tipo Adafruit_SSD1306 con las caracteristicas de la
    pantalla
81 Adafruit_SSD1306 display(SCREEN_WIDTH, SCREEN_HEIGHT, &Wire, OLED_RST);
82
83 // Esta funcion inserta variable 'var' de tipo String (con los valores de cada
    deposito) en los placeholder del HTML
84 String processor(const String& var){
85
86     if(var == "PLASTICO"){
87         return nivel[0];//si el placeholder es PLASTICO devuelve el nivel del deposito
            del plastico 'nivel[0]'
88     }
89     else if(var == "CARTON/PAPEL"){
90         return nivel[1];
91     }
92     else if(var == "METAL"){
93         return nivel[2];
94     }
95
96     else if(var == "VIDRIO"){
97         return nivel[3];
98     }
99     else if(var == "ORGANICO"){
100        return nivel[4];
101    }
102
103
104    else if(var == "TIMESTAMP"){
105        return timestamp;
106    }
107    else if (var == "RRSI"){
108        return String(rssi);
109    }
110    return String();
111}
112
113 //Inicializa la pantalla OLED
```

```
114 void startOLED(){
115     //resetea el OLED
116     pinMode(OLED_RST, OUTPUT);
117     digitalWrite(OLED_RST, LOW);
118     delay(20);
119     digitalWrite(OLED_RST, HIGH);
120
121     //inicializa el OLED
122     Wire.begin(OLED_SDA, OLED_SCL);
123     if(!display.begin(SSD1306_SWITCHCAPVCC, 0x3c, false, false)) { // Address 0x3C for
124         128x32
125         Serial.println(F("SSD1306 allocation failed"));
126         for(;); // Don't proceed, loop forever
127     }
128     display.clearDisplay(); //limpia la pantalla
129     display.setTextColor(WHITE); //establece como blanco el color de escritura
130     display.setTextSize(1); //establece el tamaño del texto en una fila
131     display.setCursor(0,0); //pone el cursor en el origen (parte superior izquierda)
132     display.print("Receptor LoRa");//una vez inicializado escribe en la pantalla este
133     mensaje
134 }
135
136 //Inicializa el modulo LoRa
137 void startLoRA(){
138     int counter;
139
140     SPI.begin(SCK, MISO, MOSI, SS);//establece los pines correspondientes
141
142     LoRa.setPins(SS, RST, DIO0); //establece el modulo LoRa
143
144     //hace una prueba para comprobar si se establece conexión en un tiempo determinado
145     while (!LoRa.begin(BAND) && counter < 10) { //envía hasta 10 paquetes mientras no
146         haya establecido conexión
147         Serial.print(".");
148         counter++;
149         delay(500);
150     }
151     if (counter == 10) {//si se han enviado los 10 paquetes quiere decir que se agoto
152         el tiempo de conexión
153         Serial.println("Error: fallo de inicializacion"); //salta un error
154     }
155     Serial.println("Inicializa LoRa OK!"); //si se establecio conexión en la pantalla
156     se muestra este mensaje
157     display.setCursor(0,10); //desplaza el cursor 10 columnas a la derecha
158     display.clearDisplay(); //limpia la pantalla
159     display.display(); //lo muestra
160     delay(2000);
161
162     //funcion para la conexión wifi
163     void connectWiFi(){
164
165         Serial.print("Conectandose a "); //indica a que WIFI se conecta
166         Serial.println(ssid);
167         WiFi.begin(ssid, contrasena); //inicia el intento de conexión al wifi con los datos
168         proporcionados
169         delay(2000);
170
171         while (WiFi.status() != WL_CONNECTED) { //realiza prueba de conexión al wifi
```

```
168     delay(500);
169     Serial.print(".");
170 }
171 // Print local IP address and start web server
172 Serial.println("");
173 Serial.println("WiFi conectado.");
174 Serial.println("IP address: ");
175 Serial.println(WiFi.localIP());//muestra el IP que se le asignó al ESP32 a través
176 // del monitor serial
176 display.setCursor(0,20); //muestra el cursor a la derecha
177 display.print("Accede a la página web: ");
178 display.setCursor(0,30);
179 display.print(WiFi.localIP());//muestra el IP que se le asignó al ESP32 para que
180 // el usuario lo introduzca en la APP
180 display.display();
181 }
182
183 // Recibe el paquete del emisor LoRa
184 void getLoRaData() {
185     Serial.print("Paquete LoRa recibido: "); //indica que ha recibido un paquete
186     // lee el paquete
187     while (LoRa.available()) {
188         String LoRaData = LoRa.readString();
189
190         Serial.print(LoRaData);
191
192         //extrae la posición del string LoRaData en el que se encuentra cada uno de los
193         // marcadores que dividen los diferentes datos dentro del mensaje
193         int pos1 = LoRaData.indexOf('/');
194         int pos2 = LoRaData.indexOf('&');
195
196
197         //extrae los datos del mensaje que están divididos por los marcadores
197         // anteriormente explicados
198         readingID = LoRaData.substring(0, pos1); //lee desde el origen a la posición 1 y
198         // lo guarda en la variable readingID
199         material = LoRaData.substring(pos1 +1, pos2).toInt(); //recibe el residuo que se
199         // esta clasificando (su posición en el vector nivel)
200         nivel[material] = LoRaData.substring(pos2+1, LoRaData.length());
201
202     }
203     //muestra el RSSI (received signal strength indicator)
204     rssI = LoRa.packetRssi();
205     Serial.print(" con RSSI ");
206     Serial.println(rssI);
207 }
208
209 // función para obtener la fecha y la hora
210 void getTimeStamp() {
211     while(!timeClient.update()) {
212         timeClient.forceUpdate();
213     }
214
215     formattedDate = timeClient.getFormattedDate(); //formatea la fecha y hora
216     Serial.println(formattedDate); //lo muestra
217
218     // Extrae la fecha
219     int splitT = formattedDate.indexOf("T"); //dividimos el string hasta la letra T
220     dia = formattedDate.substring(0, splitT); //guarda el día
221     Serial.println(dia);
```

```
222 // Extrae el tiempo
223 hora = formattedDate.substring(splitT+1, formattedDate.length()-1);
224 Serial.println(hora);
225 timestamp = dia + " " + hora;
226 }
227
228 void setup() {
229 // Initialize Serial Monitor
230 Serial.begin(115200); //inicia la comunicacion serial a una tasa de 115200 baudios
231 startOLED(); //inicializa el OLED
232 startLoRA(); //inicializa el LoRa
233 connectWiFi(); //conecta a la red WIFI
234
235 if(!SPIFFS.begin()){ //inicia el SPIFFS donde se almacena el html y la imagen
236   Serial.println("An Error has occurred while mounting SPIFFS"); //si hay
237 cualquier error se indica
238   return;
239 }
240
241 server.on("/", HTTP_GET, [](AsyncWebRequest *request){//en la pagina
242   principal
243     request->send(SPIFFS, "/index.html", String(), false, processor); // se muestra
244   el index.html y se llama a la funcion processor para obtener los datos
245 });
246 server.on("/nivel_carton", HTTP_GET, [](AsyncWebRequest *request){ //en el
247   nivel_carton del HTML
248   request->send_P(200, "text/plain", nivel[1].c_str()); //se introducel el valor
249   del deposito del carton
250 });
251 server.on("/nivel_metal", HTTP_GET, [](AsyncWebRequest *request){
252   request->send_P(200, "text/plain", nivel[2].c_str());
253 });
254 server.on("/nivel_organico", HTTP_GET, [](AsyncWebRequest *request){
255   request->send_P(200, "text/plain", nivel[4].c_str());
256 });
257 server.on("/nivel_plastico", HTTP_GET, [](AsyncWebRequest *request){
258   request->send_P(200, "text/plain", nivel[0].c_str());
259 });
260 server.on("/nivel_vidrio", HTTP_GET, [](AsyncWebRequest *request){
261   request->send_P(200, "text/plain", nivel[3].c_str());
262 });
263 server.on("/timestamp", HTTP_GET, [](AsyncWebRequest *request){
264   request->send_P(200, "text/plain", timestamp.c_str());
265 });
266 server.on("/rssи", HTTP_GET, [](AsyncWebRequest *request){
267   request->send_P(200, "text/plain", String(rssi).c_str());
268 });
269 server.on("/smartbin", HTTP_GET, [](AsyncWebRequest *request){
270   request->send(SPIFFS, "/smartbin.jpg", "image/jpg");
271 });
272
273 timeClient.begin(); // Inicia NTPClient para obtener el tiempo
274
275 timeClient.setTimeOffset(3600); //para ajustar el tiempo a la zona horaria de
276 //españa (GMT +1) se aplica un offset de 3600
277 }
```

```
276 void loop() {  
277     //analiza si hay paquetes por recibir  
278     int packetSize = LoRa.parsePacket();  
279     if (packetSize) {  
280         getLoRaData(); //obtiene los datos del mensaje  
281         getTimeStamp(); //obtiene la fecha y la hora  
282     }  
283 }
```

**15.2.7. *Código del servidor web***

```
1 <!DOCTYPE HTML><html>
2 <head>
3   <meta name="viewport" content="width=device-width, initial-scale=1">
4   <link rel="icon" href="data:,">
5   <title>SMARTBIN DVM</title>
6
7   <script src='https://kit.fontawesome.com/a076d05399.js'></script>
8   <style>
9     body {
10       margin: 0;
11       font-family: Arial, Helvetica, sans-serif;
12       text-align: center;
13     }
14     header {
15       margin: 0;
16       padding-top: 8vh;
17       padding-bottom: 5vh;
18       overflow: hidden;
19       background-image: url(smartbin.jpg);
20       background-size: cover;
21       color: black;
22     }
23     h2 {
24       font-size: 2.0rem;
25     }
26     p { font-size: 1.2rem; }
27     .units { font-size: 1.2rem; }
28     .readings { font-size: 2.0rem; }
29   </style>
30 </head>
31 <body>
32   <header>
33     <h2>SMARTBIN DVM</h2>
34     <p><strong>Último paquete recibido:<br/><span id="timestamp">%TIMESTAMP%</span></strong></p>
35     <p>LoRa RSSI: <span id="rssI">%RSSI%</span></p>
36   </header>
37   <main>
38
39     <p>
40       <i style='font-size:24px' class='far'>#xf15b;</i> Nivel Carton/Papel: <span
41       id="nivel_carton" class="readings">%NIVEL_CARTON%</span>
42       <sup>#37;</sup>
43     </p>
44     <p>
45       <i style='font-size:24px' class='fas'>#xf076;</i> Nivel Metal: <span
46       id="nivel_metal" class="readings">%NIVEL_METAL%</span>
47       <sup>#37;</sup>
48     </p>
49     <p>
50       <i style='font-size:24px' class='fas'>#xf0c3;</i> Nivel Vidrio: <span
51       id="nivel_vidrio" class="readings">%NIVEL_VIDRIO%</span>
52       <sup>#37;</sup>
53     </p>
54     <p>
55       <i style='font-size:24px' class='fas'>#xf5d1;</i> Nivel Organico: <span
```

```
56     <i class="fas fa-angle-double-down" style="color:#e8c14d;"></i> Nivel Plastico:  
57     <span id="nivel_plastico" class="readings">%NIVEL_PLASTICO%</span>  
58     <sup>#37;</sup>  
59   </p>  
60 </main>  
61 <script>  
62  
63  
64  
65 setInterval(updateValues, 10000, "nivel_carton");  
66 setInterval(updateValues, 10000, "nivel_metal");  
67 setInterval(updateValues, 10000, "nivel_vidrio");  
68 setInterval(updateValues, 10000, "nivel_organico");  
69 setInterval(updateValues, 10000, "nivel_plastico");  
70 setInterval(updateValues, 10000, "rss");  
71 setInterval(updateValues, 10000, "timestamp");  
72  
73  
74  
75 function updateValues(value) {  
76     var xhttp = new XMLHttpRequest();  
77     xhttp.onreadystatechange = function() {  
78         if (this.readyState == 4 && this.status == 200) {  
79             document.getElementById(value).innerHTML = this.responseText;  
80         }  
81     };  
82     xhttp.open("GET", "/" + value, true);  
83     xhttp.send();  
84 }  
85 </script>  
86 </body>  
87 </html>
```

**15.2.8. *Código de la aplicación móvil***

```
1 package com.example.smartbin;
2
3 import android.content.Intent;
4 import android.os.Bundle;
5
6 import com.google.android.material.floatingactionbutton.
7 FloatingActionButton;
8 import com.google.android.material.snackbar.Snackbar;
9
10 import androidx.appcompat.app.AppCompatActivity;
11 import androidx.appcompat.widget.Toolbar;
12
13 import android.view.View;
14
15 import android.view.Menu;
16 import android.view.MenuItem;
17 import android.widget.Button;
18 import android.widget.EditText;
19 import android.widget.TextView;
20
21 public class MainActivity extends AppCompatActivity {
22     EditText e_ip;
23     Button b_entrar;
24     TextView t_mensaje;
25
26     @Override
27     protected void onCreate(Bundle savedInstanceState) {
28         setContentView(R.layout.activity_main);
29         e_ip = (EditText) findViewById(R.id.edit_IP);
30         b_entrar = (Button) findViewById(R.id.boton_entrar);
31     }
32
33     //Toolbar toolbar = findViewById(R.id.toolbar);
34     // setSupportActionBar(toolbar);
35
36     // FloatingActionButton fab = findViewById(R.id.fab);
37     // fab.setOnClickListener(new View.OnClickListener()
38     () {
39
40
41     }
42 }
```

```
43
44     public void boton_entrar_onClick(View v) {
45         if ( e_ip.getText().toString().substring(0,3).
46             equals("192")) {
47             t_mensaje.setText("Cargando datos...");
48             Intent intent = new Intent(this,
49             activity_lora_web.class);
50             intent.putExtra("ip",e_ip.getText().toString
51             ());
52             startActivity(intent);
53         }
54
55
56     @Override
57     public boolean onCreateOptionsMenu(Menu menu) {
58         // Inflate the menu; this adds items to the action
59         // bar if it is present.
60         getMenuInflater().inflate(R.menu.menu_main, menu);
61         return true;
62     }
63
64     @Override
65     public boolean onOptionsItemSelected(MenuItem item) {
66         // Handle action bar item clicks here. The action
67         // bar will
68         // automatically handle clicks on the Home/Up
69         // button, so long
70         // as you specify a parent activity in
71         // AndroidManifest.xml.
72         int id = item.getItemId();
73
74         //noinspection SimplifiableIfStatement
75         if (id == R.id.action_settings) {
76             return true;
77         }
78
79         return super.onOptionsItemSelected(item);
80     }
81 }
```

```
1 <?xml version="1.0" encoding="utf-8"?>
2 <androidx.coordinatorlayout.widget.CoordinatorLayout xmlns:
3     android="http://schemas.android.com/apk/res/android"
4     xmlns:app="http://schemas.android.com/apk/res-auto"
5     xmlns:tools="http://schemas.android.com/tools"
6     android:id="@+id/TextView"
7     android:layout_width="match_parent"
8     android:layout_height="match_parent"
9     tools:context=".MainActivity">
10
11    <com.google.android.material.appbar.AppBarLayout
12        android:layout_width="match_parent"
13        android:layout_height="wrap_content"
14        android:theme="@style/AppTheme.AppBarOverlay" />
15
16    <androidx.constraintlayout.widget.ConstraintLayout
17        android:id="@+id/constraintLayout"
18        android:layout_width="match_parent"
19        android:layout_height="match_parent">
20
21        <EditText
22            android:id="@+id/edit_IP"
23            android:layout_width="wrap_content"
24            android:layout_height="wrap_content"
25            android:ems="10"
26            android:inputType="textPersonName"
27            android:text="Introduzca la IP"
28            app:layout_constraintBottom_toBottomOf="parent"
29            app:layout_constraintEnd_toEndOf="parent"
30            app:layout_constraintHorizontal_bias="0.497"
31            app:layout_constraintStart_toStartOf="parent"
32            app:layout_constraintTop_toTopOf="parent"
33            app:layout_constraintVertical_bias="0.361" />
34
35        <TextView
36            android:id="@+id/textView"
37            android:layout_width="wrap_content"
38            android:layout_height="wrap_content"
39            android:layout_marginTop="7dp"
40            app:layout_constraintEnd_toEndOf="parent"
41            app:layout_constraintStart_toStartOf="parent"
42            app:layout_constraintTop_toBottomOf="@+id/
43                edit_IP" />
44        </androidx.constraintlayout.widget.ConstraintLayout>
45
46        <Button
```

```
45      android:id="@+id/boton_entrar"
46      android:layout_width="wrap_content"
47      android:layout_height="wrap_content"
48      android:onClick="boton_entrar_onClick"
49      android:text="Entrar"
50      app:layout_anchor="@+id/constraintLayout"
51      app:layout_anchorGravity="center" />
52
53
54 </androidx.coordinatorlayout.widget.CoordinatorLayout>
```

```
1 package com.example.smartbin;
2
3 import androidx.appcompat.app.AppCompatActivity;
4
5 import android.os.Bundle;
6 import android.webkit.WebSettings;
7 import android.webkit.WebView;
8 import android.webkit.WebViewClient;
9
10 public class activity_lora_web extends AppCompatActivity {
11
12     @Override
13     protected void onCreate(Bundle savedInstanceState) {
14         super.onCreate(savedInstanceState);
15         setContentView(R.layout.activity_lora_web);
16         String ip = getIntent().getStringExtra("ip");
17         ip = "http://" +ip ;
18
19         WebView web = (WebView) findViewById(R.id.webview);
20         web.setWebViewClient(new MyWebViewClient());
21         WebSettings settings =web.getSettings();
22         settings.setJavaScriptEnabled(true);
23         web.loadUrl(ip);
24
25     }
26     private class MyWebViewClient extends WebViewClient {
27         public boolean shouldOverrideUrlLoading (WebView
28             view, String ip){
29             view.loadUrl(ip);
30             return true;
31         }
32     }
33 }
```

```
1 <?xml version="1.0" encoding="utf-8"?>
2 <androidx.constraintlayout.widget.ConstraintLayout xmlns:
  android="http://schemas.android.com/apk/res/android"
3   xmlns:app="http://schemas.android.com/apk/res-auto"
4   xmlns:tools="http://schemas.android.com/tools"
5   android:layout_width="match_parent"
6   android:layout_height="match_parent"
7   tools:context=".activity_lora_web">
8
9
10 <WebView
11   android:id="@+id/webview"
12   android:layout_width="417dp"
13   android:layout_height="567dp"
14   app:layout_constraintBottom_toBottomOf="parent"
15   app:layout_constraintEnd_toEndOf="parent"
16   app:layout_constraintStart_toStartOf="parent"
17   app:layout_constraintTop_toTopOf="parent"
18   app:layout_constraintVertical_bias="1.0" />
19 </androidx.constraintlayout.widget.ConstraintLayout>
```