

Informe final

Control a distancia de un vehículo explorador móvil con relevamiento en servidor de imágenes sincronizadas con información de sensor de proximidad.



**UNIVERSIDAD
NACIONAL
DE LA PLATA**

Práctica Profesional Supervisada - Ingeniería en computación

Alumnos

Nº de legajo

Dehan Lucas

565/1

Duarte Victor

1055/7



Centro de Técnicas Analógico-Digitales (CETAD)

ÍNDICE

Introducción	1
Objetivos del proyecto	2
Objetivo Principal	2
Objetivo Secundario	4
Análisis de requerimientos	5
Requerimientos funcionales	5
Requerimientos no funcionales	5
Diseño del hardware	6
EDU-CIAA	6
Puente H	7
Controlador ULN2003A	9
Sensor LiDAR TF-LC02	10
ESP32-CAM	11
Alimentación del sistema	14
Diseño del PCB	16
Agujero pasante	16
Pads	17
Reglas del diseño	17
Conectores	18
Puente H	19
Controlador ULN2003A	20
Sensor LiDAR TF LC-02	20
ESP32-CAM	21
Conector IDC10M-90	22
Conector alimentación ESP32-CAM	22
Diseño del software	25
EDU-CIAA	26
ctrl_marchas	27
esp32_cam	28
mensajes	29



Centro de Técnicas Analógico-Digitales (CETAD)

main	29
ESP32-CAM	30
conexion_udp	30
camara	32
pins_camara	33
motor_28byj48	33
tf_lc02	35
ftp	36
esp32_servidor_ciaa	36
Snaprunner	39
Ensayos y mediciones	42
Dificultades en proceso de diseño	46
Conclusiones	49
Tiempo dedicado y Presupuesto final	50
Bibliografía	52
Anexos	54
Esquemático.	54
PCB	55
Agradecimientos	57



Centro de Técnicas Analógico-Digitales (CETAD)

Introducción

En el Centro de Técnicas Analogico-Digitales (CeTAD-GCA, LEICI), como proyecto tecnológico para la facultad de ingeniería, surgió la idea de realizar un vehículo móvil el cual pueda obtener imágenes en su recorrido con el objetivo de entrenar a futuro una inteligencia artificial la cual pueda automatizar la conducción del vehículo teniendo referencia de las distancias para la navegación.

Este proyecto se basa en construir un Robot Móvil de Exploración (RME), el cual será controlado a través de una aplicación móvil mediante una conexión inalámbrica. Durante su desplazamiento, el robot capturará imágenes y registrará mediciones del entorno a través de sensores especializados.

Aprovechando los avances en **comunicaciones inalámbricas**, se desarrolla un sistema capaz de enviar imágenes enriquecidas con datos obtenidos de sensores de proximidad a un servidor local. Esto permite la creación de un conjunto de datos estructurado para el entrenamiento de modelos de inteligencia artificial orientados a la conducción autónoma.

Como parte de la **Práctica Profesional Supervisada**, se fabrica un prototipo funcional del robot y una aplicación móvil con una interfaz gráfica para su control remoto y adquisición de datos. El diseño del sistema incluye un **chasis con cuatro ruedas**, un circuito electrónico para la movilidad, un **sensor de proximidad montado sobre un motor paso a paso** para el escaneo del entorno, una cámara para la captura de imágenes y una conexión inalámbrica para la transmisión de datos al servidor.



Centro de Técnicas Analógico-Digitales (CeTAD)

Objetivos del proyecto

Objetivo Principal

El proyecto consta de dos objetivos principales:

- Desarrollar un RME y una aplicación móvil para controlarlo a distancia.
- Capturar imágenes y datos de un módulo LiDAR sincronizados en cada barrido, y enviarlos a un servidor.

Se realiza un prototipo a escala como prueba de concepto que sea verificable en el ámbito del CeTAD con la placa EDU-CIAA y un ESP32-CAM de un sistema capaz de realizar distintas tareas como:

1. Ensamblaje del kit del RME con sus respectivos sensores.
2. Diseñar una aplicación móvil con Flutter, para la navegación a distancia y envío de datos .
3. Para la comunicación inalámbrica con la aplicación, se usará un módulo wifi del ESP32 que será intermediario entre la aplicación móvil y la Edu-Ciaa. Para esta comunicación se implementará el protocolo UDP debido a su mayor velocidad de transmisión de datos.
4. Para la comunicación con el servidor, se usará también el módulo wifi del ESP32 implementando el protocolo FTP.
5. Para obtener las imágenes, se utilizará la cámara OV2640 integrada en el ESP32-CAM.
6. Para las mediciones de proximidad se utilizará un sensor LiDAR, el cual estará montado sobre un motor paso a paso y tiene una comunicación con el ESP32.



Centro de Técnicas Analógico-Digitales (CETAD)

Con estos objetivos en mente podemos hacer un esquema con el EDU-CIAA y el Módulo ESP32-CAM (naylampmechatronics, 2025) conectado a los siguientes componentes:

- Una cámara integrada en el módulo ESP32 para tomar imágenes.
- Un módulo de desarrollo Puente H (naylampmechatronics, 2025), para controlar los motores de movimiento (naylampmechatronics, 2025).
- Un sensor LiDAR (LAB1 Technologies, 2025), para obtener medidas de proximidad.
- Un módulo Controlador (Todomicro, 2025) para controlar el motor paso a paso (naylampmechatronics, 2025).
- Una Aplicación Móvil (inalámbrica), para control del usuario
- Un Servidor FTP (inalámbrica), para almacenamiento de imágenes.

En la siguiente imagen (Ver Figura 1) se observa una representación del esquema mencionado anteriormente.

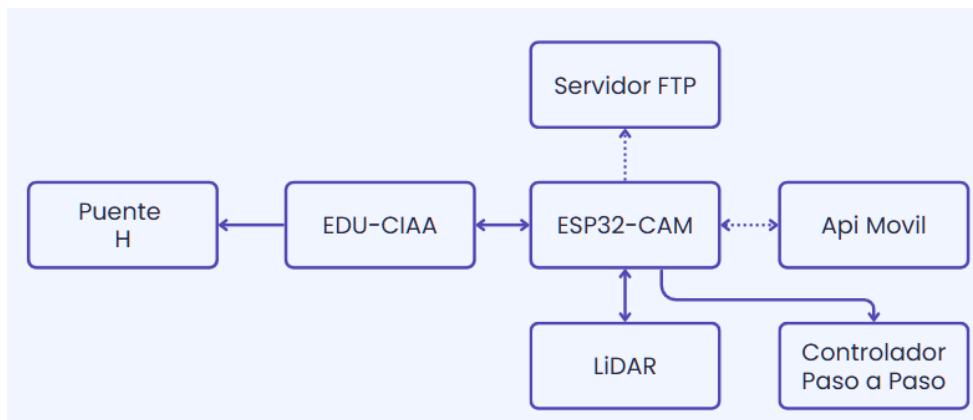


Figura 1: Diagrama en bloques del sistema.

Como se puede observar en la imagen, a la EDU-CIAA, del lado izquierdo podemos ver mediante una conexión unidireccional se conecta el puente-h que emitirá la señal hacia los motores para avance y retroceso.

Luego una conexión bidireccional con el ESP32, el cual actuará de intercomunicador, se encargará de las demás conexiones. Esto quiere decir que enviará a la EDU-CIAA todas las señales recibidas de distintas conexiones.



Centro de Técnicas Analógico-Digitales (CETAD)

Las demás conexiones serán desde el ESP32, las conexiones son:

Unidireccionales:

- Servidor FTP al cual se le enviarán las imágenes tomadas.
- Controlador Paso a Paso, el cual enviará la señal al motor paso a paso para realizar una barrido

Bidireccionales:

- Sensor LiDAR, enviará una señal para que realice una medición y retorne la misma.
- Aplicación Móvil, enviará información de la cámara, y recibirá comandos de control.

Objetivo Secundario

Con vista al futuro la idea es poder implementar interrupciones, para poder utilizar funciones como sleep tanto en la EDU-CIAA como en el ESP32-CAM para un ahorro energético.

Para realizar el entrenamiento de la inteligencia artificial de forma más efectiva, se podría clasificar las imágenes al subirlas al servidor FTP dependiendo si la imagen se toma cuando el RME está en dirección recta, o cuando está girando con su respectiva dirección de giro.

Sensar el nivel de batería del dispositivo para detectar la carga de la batería en porcentaje.



Centro de Técnicas Analógico-Digitales (CETAD)

Análisis de requerimientos

Requerimientos funcionales

- Conducción del RME a distancia.
- Proporcionar un Access Point para conexión de la aplicación de control.
- Captura de imágenes
- Barrido con el sensor LiDAR para detección de objetos en el área de imagen capturada.
- Enriquecimiento de imagen capturada con los datos obtenidos del barrido.
- Capacidad de conectarse a una red wifi existente para subir las imágenes al servidor FTP.

Requerimientos no funcionales

- Utilización de la placa EDU-CIAA.
- Utilización de la placa ESP32-CAM para la transmisión Wifi y Access Point.
- Utilización del protocolo UDP para comunicación ESP32 con Api Móvil.
- Utilización del protocolo FTP para almacenar imágenes.
- Programación de software en lenguaje C mediante las herramientas proporcionadas por el Proyecto CIAA.
- Programación de software en lenguaje C++ mediante Arduino IDE con librerías proporcionadas para desarrollo en placas ESP32.
- Programación de software en lenguaje Dart mediante la herramienta Flutter para realizar la aplicación móvil.
- Desarrollo del PCB con los circuitos necesarios para conectar los sensores.
- Realizar el prototipo final para su presentación.



Centro de Técnicas Analógico-Digitales (CETAD)

Diseño del hardware

En base al diagrama en bloques explicado en la sección “Objetivos del proyecto” se diseñaron dos esquemáticos en el programa Proteus, uno del conexionado en la EDU-CIAA y el otro con el conexionado del ESP32-CAM.

EDU-CIAA

Elegimos para este proyecto trabajar con la placa EDU-CIAA para controlar el movimiento del RME ya que ofrece varias características de utilidad para el desarrollo:

- Procesador Dual Core de 32-bit, lo que permite procesar los comandos más rápido y conseguir un mejor manejo.
- Posee diversos puertos de comunicación: USB, RS232, I2C, UART. En particular, la EDU-CIAA tiene 3 puertos de UART lo cual fue útil para la depuración.
- Tiene salidas de alimentación de 5V y 3.3V para alimentar componentes.
- Se pueden fabricar y conectar “Ponchos”. Un “Poncho” es un módulo diseñado para conectarse sobre placas CIAA y puede agregar interfaces de comunicación, sensores, actuadores, fuentes de alimentación o borneras de conexión.

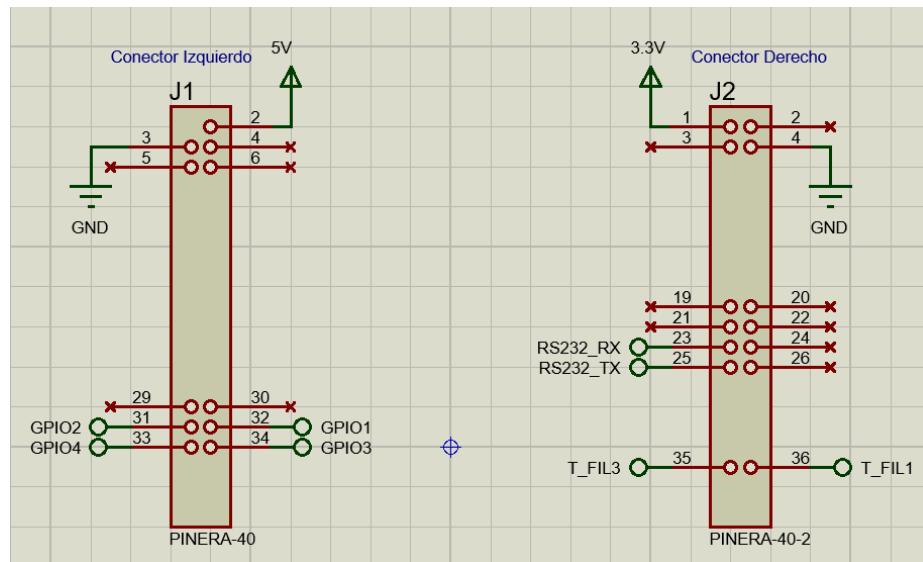


Figura 2: Esquemático de conexiones a EDU-CIAA.



Centro de Técnicas Analógico-Digitales (CETAD)

La imagen anterior es una representación de los conectores (J1 y J2) que irán encastrados en los conectores de la EDU-CIAA en Proteus. Cada conector contiene pines que pueden conectarse a otros dispositivos como sensores, pulsadores, LEDs, etc. Aquellos pines que serán utilizados para una comunicación con otros módulos están representados con un símbolo en color verde, junto con el nombre del pin en la EDU-CIAA y aquellos que no serán utilizados en este proyecto serán eliminados de la pinera para ganar espacio, para que pasen pistas o serán marcados con otro símbolo que indica que no está conectado a ningún otro lado pero se deja el pin físico para usarse como sujeción mecánica del PCB a la placa. (ver bibliografía EDU-CIAA-NXP).

A continuación se describen los distintos componentes del hardware conectados a la EDU-CIAA y/o al ESP32-CAM y sus conexiones correspondientes:

Puente H

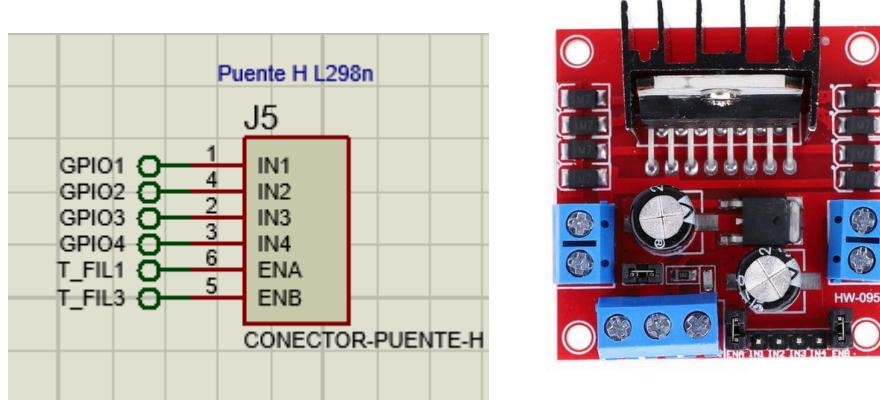


Figura 3: Diagrama del conector del Puente H (izquierda), imagen del Puente H (derecha).

Para este proyecto se utilizó el módulo HW-095, que utiliza el driver L298N. Este módulo soporta la alimentación de dos motores de corriente continua o DC independientes, comúnmente llamados Motor A que se conecta a la bornera izquierda del módulo, y Motor B que se conecta a la bornera derecha. Los pines IN1 e IN2 controlan el Motor A, y los pines



Centro de Técnicas Analógico-Digitales (CETAD)

IN3 e IN4 controlan el Motor B, los pines ENA y ENB se utilizan para controlar la velocidad de los motores con una señal PWM en los motores A y B respectivamente.

En el esquemático, se observa que la comunicación entre el Puente H y la EDU-CIAA se realizará por los siguientes pines: GPIO1 y GPIO2 para controlar el Motor A, GPIO3 y GPIO4 para controlar el Motor B, los pines T_FIL1 y T_FIL3 se utilizan para enviar señales PWM y controlar la velocidad del RME.

El puente H está conectado a motores de corriente continua diseñados para proyectos de robótica.

Originalmente el chasis venía con dos motores en la parte delantera del vehículo junto con una rueda de giro libre atrás, pero con el avance del desarrollo observamos que la rueda de giro libre era la causa de varios inconvenientes:

- Cuando la rueda no está centrada, hace más difícil que gire en sentido opuesto al que está la rueda.
- No gira bien en superficies no lisas.
- Se ensucia y enreda con facilidad y es difícil de limpiar.

Por esos motivos, se cambió la rueda de giro libre por otros dos motores uno de cada lado conectados cada uno en paralelo con el motor de su mismo lado que ya estaba instalado, para poder utilizar un mismo puente H para girar las cuatro ruedas. Dando como resultado final la siguiente configuración:

- Motor A: Controla los motores izquierdos del RME.
- Motor B: Controla los motores derechos del RME.

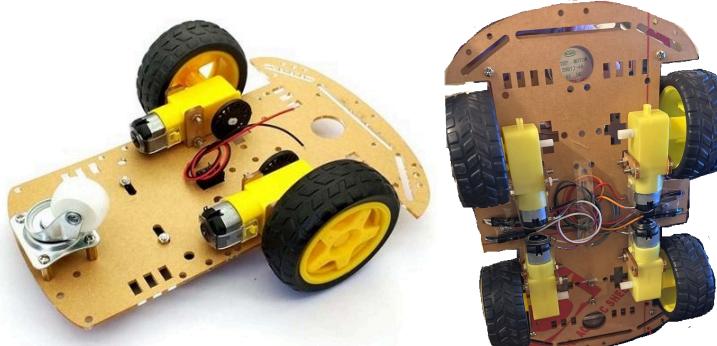


Figura 4: Ejemplo de chasis con y sin rueda de giro libre.



Centro de Técnicas Analógico-Digitales (CETAD)

Controlador ULN2003A

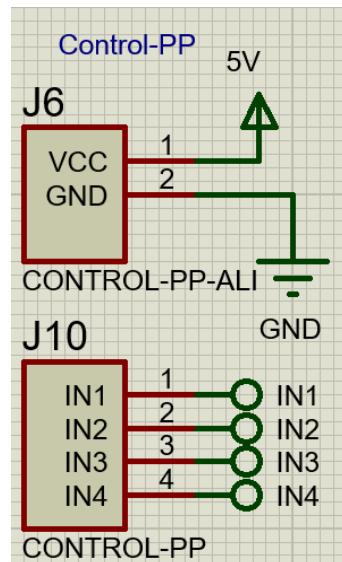


Figura 5: Diagrama del controlador (izquierda), imagen del controlador (derecha).

Para utilizar el motor paso a paso, se necesita de un circuito electrónico con transistores o conectar el motor a un controlador por lo que se eligió implementar la segunda opción. El controlador estará alimentado por la EDU-CIAA con una tensión de 5V. Los pinos IN1 al IN4 se usan para enviar señales para mover al motor, estos pinos están conectados al ESP32-CAM. El ULN2003A tiene un conector JST-XH de 5 pines para conectar el motor paso a paso, que para este controlador es el modelo 28BYJ-48, que se observa en la figura a continuación.



Figura 6: Motor paso a paso 28BYJ-48.



Centro de Técnicas Analógico-Digitales (CETAD)

Sensor LiDAR TF-LC02

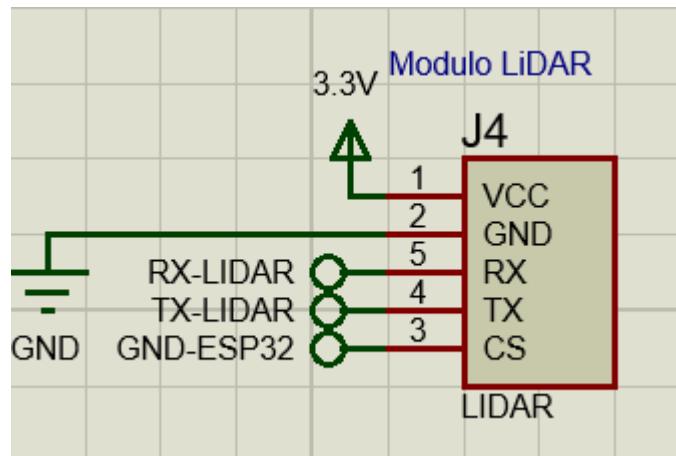


Figura 7: Diagrama del sensor LiDAR.

El sensor LiDAR con el que trabajamos posee las siguientes características:

- Rango de detección: 3-200 cm
- Tiempo de adquisición de datos: 33 ms
- Exactitud de $\pm 2\text{cm}$ detectando objetos en un rango hasta 100cm, $\pm 5\%$ a distancias superiores a ese rango.
- Comunicación UART.

Para más detalles ver bibliografía (LAB1 Technologies ,2025).

El sensor está alimentado por la EDU-CIAA con una tensión de 3.3V. El pin CS tiene que estar conectado a GND para habilitar la comunicación por UART, y la comunicación UART se hace con el ESP32-CAM. El sensor tiene un conector JST-SH de 6 pines de 1mm, pero uno de ellos es NC (Not-Connected), por lo que no se toma en cuenta para las conexiones.



Centro de Técnicas Analógico-Digitales (CETAD)

ESP32-CAM

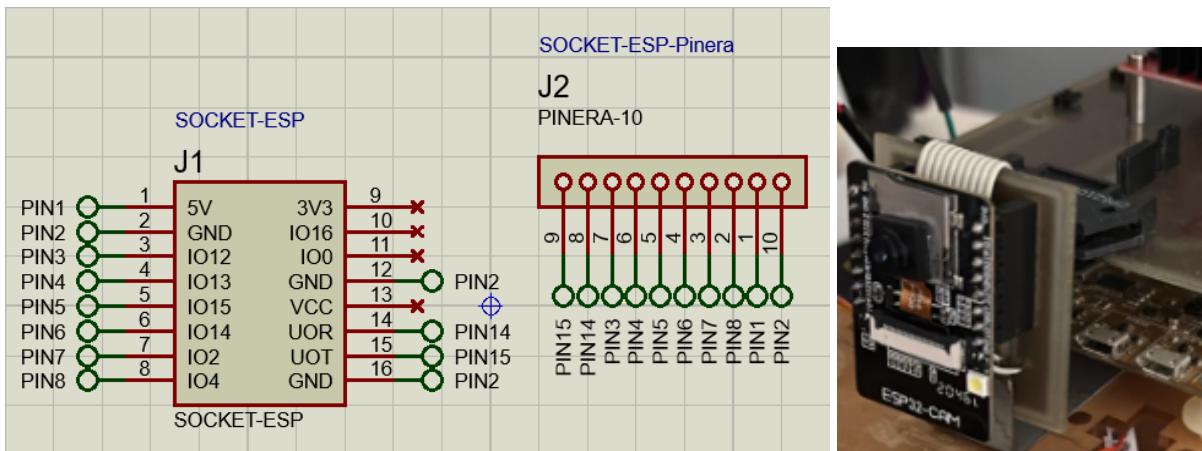


Figura 8: Diagrama del ESP32-CAM.

El módulo ESP32-CAM que tiene una cámara integrada la cual es utilizada para la captura de imágenes, para conectividad posee WiFi y Bluetooth, se decidió utilizar AP ya que tiene mayor alcance para la conexión de la api móvil y WiFi para conectarse al servidor FTP. El ESP32 posee pines GPIO que son utilizados para comunicaciones con componentes a elección.

Como se observa en la figura 8, se optó por crear un poncho también para el ESP32, el cual está conectado a la EDU-CIAA mediante un conector macho 10 pines 90° (figura 9) y un cable plano P/ci (IDC 10M-90) soldado al poncho del ESP32. El ESP32-CAM está alimentado a una tensión de 5V. Esta conexión permite colocar el ESP32-CAM horizontalmente para poder obtener imágenes mediante su cámara.



Centro de Técnicas Analógico-Digitales (CETAD)

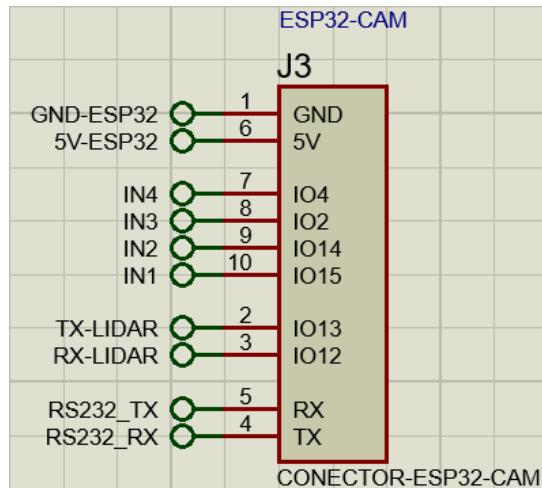


Figura 9: Diagrama del conector al ESP32-CAM en el poncho de la EDU-CIAA.

Se describen las conexiones de los componentes electrónicos a la ESP32-CAM en la siguiente tabla:

Tabla 1. Conexiones entre el módulo ESP32-CAM y los demás componentes

Componente	Pin Componente	Pin ESP32-CAM	Pinera ESP32-CAM
TF-LC02	RX	IO12	PIN3
	TX	IO13	PIN4
	CS	GND	PIN2
	NC (Not connected)	-	-



Centro de Técnicas Analógico-Digitales (CETAD)

ULN2003A	IN1	IO15	PIN5
	IN2	IO14	PIN6
	IN3	IO2	PIN7
	IN4	IO4	PIN8
EDU-CIAA	RS232_TX	UOR	PIN14
	RS232_RX	UOT	PIN15
Alimentación	5V	5V	PIN1
	GND	GND	PIN2



Centro de Técnicas Analógico-Digitales (CETAD)

Alimentación del sistema

Para alimentar el sistema, tenemos dos fuentes de alimentación con 2 baterías de 3.7V 2200mAh, por lo tanto cada fuente otorga 7.4V de tensión. Para distribuir el consumo, se utilizan las fuentes para alimentar el sistema de la siguiente manera:

- La primera fuente se utiliza para alimentar el puente H de forma directa por la entrada 6-12V.
- La segunda fuente se utiliza para alimentar la EDU-CIAA y la ESP32-CAM de forma paralela. Se optó por esta configuración, dado que la EDU-CIAA no es capaz de soportar la demanda energética del ESP32 cuando este realiza conexiones inalámbricas.

Para alimentar la EDU-CIAA por una fuente de alimentación externa, está debe ser a 5V. Mientras que la ESP32-CAM puede ser alimentada tanto a 3.3V como a 5V, por lo que se alimenta a 5V para que la alimentación sea en paralelo con la EDU-CIAA.

Para reducir la tensión de la fuente de 7,4V a 5V, la fuente se conecta a una fuente Step-Down (Hobbytronica, 2025), que se observa a continuación:



Figura 10: Fuente Step-Down LM2596 DC-DC, utilizado en el proyecto.

La EDU-CIAA alimenta componentes del sistema, por lo que se tiene que tener en cuenta que hay un fusible en ambos conectores tanto para 3.3V y 5V. Por lo tanto hay cuatro fusibles en total y el máximo de corriente que puede tolerar cada fusible es de 300 mA.



Centro de Técnicas Analógico-Digitales (CETAD)

Para asegurarnos que sea posible alimentar estos componentes sin que se produzcan daños, es necesario realizar un cálculo aproximado de la corriente necesaria para la alimentación.

Dichos cálculos son estimados en base a las hojas de datos de cada componente.

En conector J1:

Dispositivo	Consumo de corriente
Controlador PaP	5.4 mA
Motor PaP	240 mA
Total J1	245.4 mA

En conector J2:

Conecotor	Consumo de corriente
LiDAR	15 mA
Total J2	15 mA

Total de consumo:

Conecotor	Consumo de corriente
J1	245.4 mA
J2	15 mA
Total	260.4 mA



Centro de Técnicas Analógico-Digitales (CETAD)

Diseño del PCB

En base a lo explicado en la sección del esquemático, se procede a realizar el PCB. Los componentes de PCB en Proteus se miden en milímetros (mm), pulgadas (in) y mils o thou (th). Para evitar confusiones en las medidas que se mostrarán más adelante, a continuación se presenta una tabla con las unidades de medida del sistema anglosajón de unidades y su equivalencia a milímetros.

Tabla 2. Unidades de medida anglosajona y su valor en milímetros.

Unidad	valor en mm
1 in	25,4 mm
1 th	0,0254 mm

Para este proyecto se diseñaron componentes de PCB:

Agujero pasante

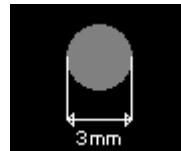


Figura 11: Diagrama la vía “3mm”

Los agujeros son utilizados para poder adherir a través de tornillos los módulos Puente H y Controlador PaP (ULN2003A) al poncho de la EDU-CIAA. Son un agujero pasante de diámetro 3 mm.



Centro de Técnicas Analógico-Digitales (CETAD)

Pads

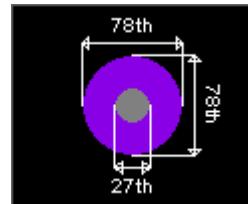


Figura 12: Diagrama del pad “M78X27”

Acorde a las sugerencias del Área Técnica de Electrónica e Instrumental (ATEI), se implementó el pad M78X27 en todos los pines del PCB.

Reglas del diseño

- Pad - Pad: 20 th.
- Pad - Trace: 20 th.
- Graphic: 20 th
- Edge: 60 th



Centro de Técnicas Analógico-Digitales (CETAD)

Conecadores

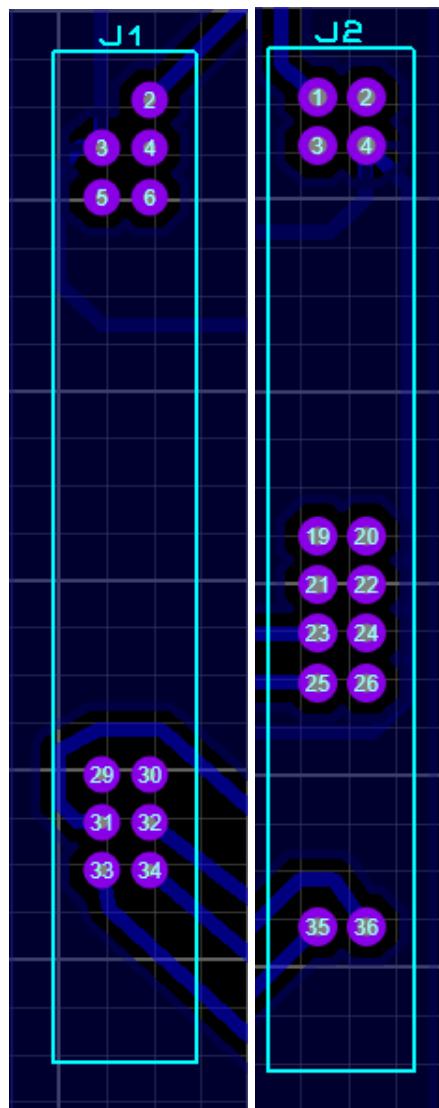


Figura 13: Diagrama las conectores en el PCB

La imagen anterior es una representación de las conectores (J1 y J2) en el PCB.



Centro de Técnicas Analógico-Digitales (CETAD)

Puente H

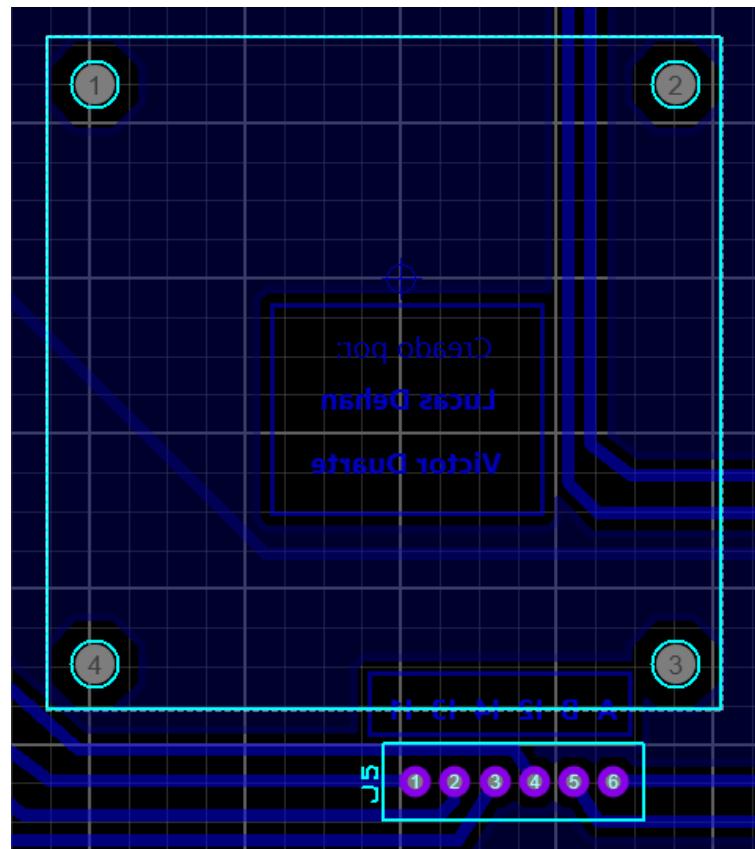


Figura 14: Diagrama del footprint del puente H junto con su conector en el PCB.
Para ahorrar espacio en el chasis, el puente H se coloca en el PCB por lo que se realizó el footprint de acuerdo a las dimensiones del módulo.



Centro de Técnicas Analógico-Digitales (CETAD)

Controlador ULN2003A

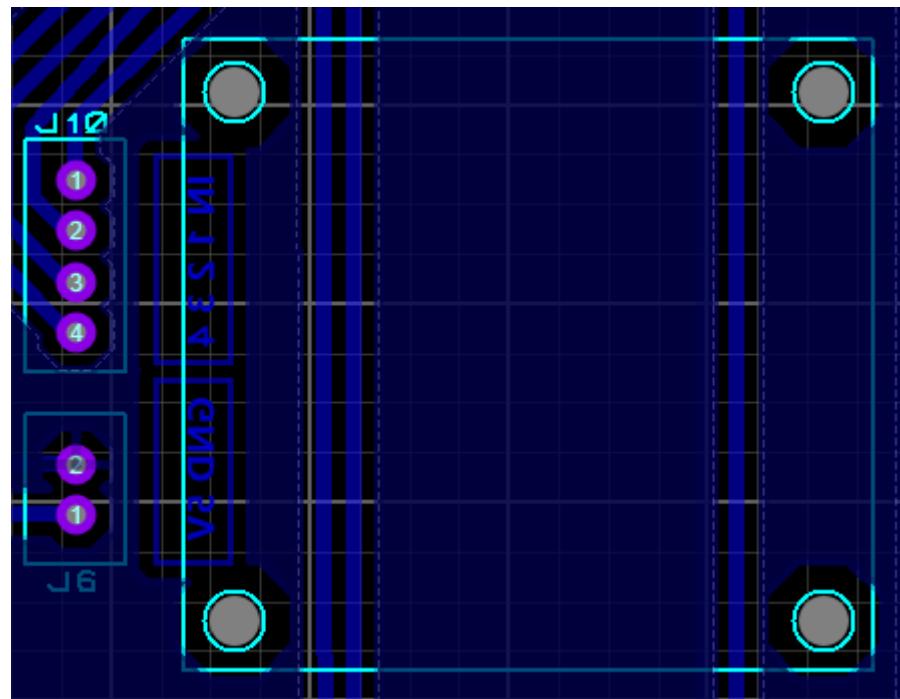


Figura 15: Diagrama del footprint del controlador junto con sus conectores en el PCB.

Se realizó el footprint de acuerdo a las dimensiones del controlador para colocarlo en la PCB. Tiene un conector para las entradas IN a IN4 y un conector para la alimentación.

Sensor LiDAR TF LC-02



Figura 16: Diagrama del conector del sensor

Se realizó el footprint del conector de acuerdo al cableado del LiDAR. El pin 1 está conectado a 3.3V, el pin 2 está conectado a GND y los pinos 3,4 y 5 conectados a Rx, Cs y Tx respectivamente.



Centro de Técnicas Analógico-Digitales (CETAD)

ESP32-CAM

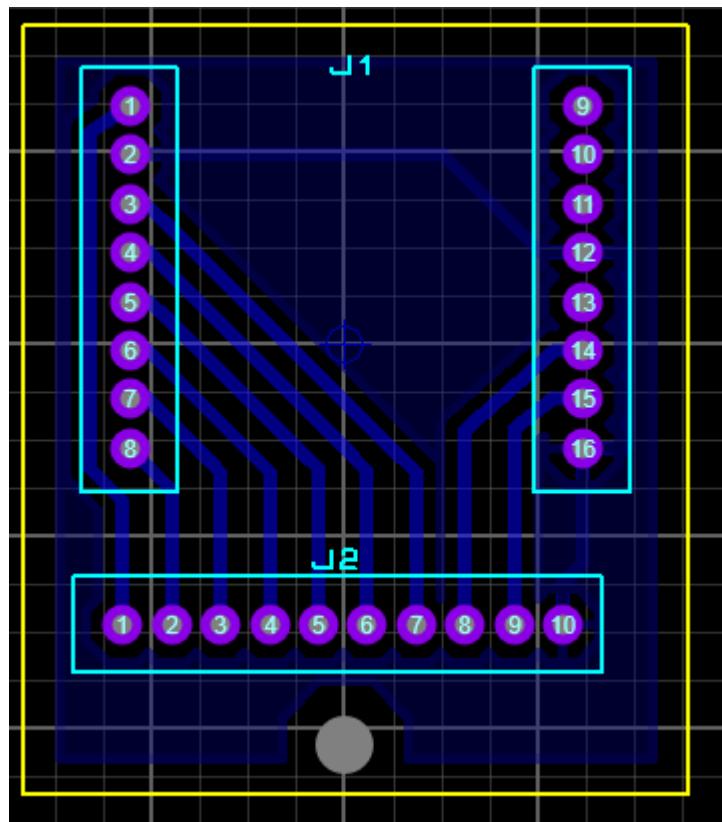


Figura 17: Diagrama del ESP32-CAM.

Se realizó el PCB que va colocado al ESP32 con un área un poco más grande que la del módulo para añadir un agujero pasante (de sujeción) y dejar una capa de cobre a los lados para interconectar el plano de tierra.



Centro de Técnicas Analógico-Digitales (CETAD)

Conecotor IDC10M-90

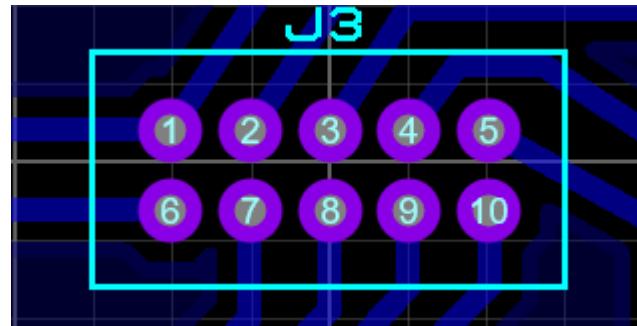


Figura 18: Diagrama del conector.

Se realizó el footprint de acuerdo a las dimensiones del conector.

Conecotor alimentación ESP32-CAM

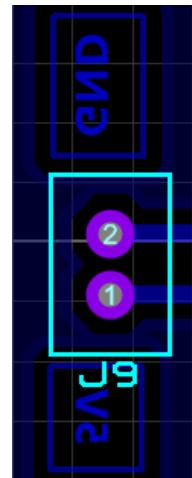


Figura 19: Diagrama del conector de alimentación.



Centro de Técnicas Analógico-Digitales (CETAD)

Una vez realizado el circuito en Proteus, el mismo fue impreso como se muestra en la siguiente imagen.

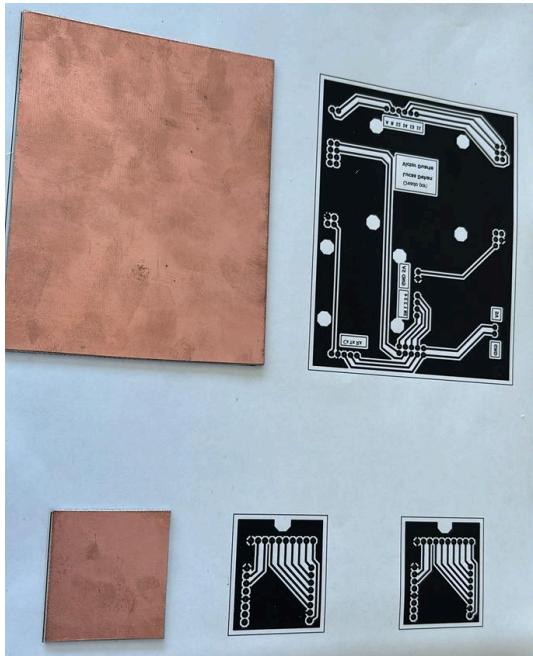


Figura 20: Impresión del circuito en papel Land.

Se recortó a medida la placa virgen de cobre monofaz de Pertinax¹, y mediante un proceso de planchado, se adhiere la tinta de la impresión a la placa la cual se utiliza para proteger el cobre que se desea mantener en la placa, dado que luego la placa es sumergida en ácido, como muestra la siguiente imagen.

¹ **Pertinax:** Papel impregnado en resina fenólica.



Centro de Técnicas Analógico-Digitales (CETAD)



Figura 21: Adhesión de tinta a la placa y aplicación de ácido.

El proceso de sumergir en ácido la placa es para que el mismo ataque al cobre expuesto, y lo corroa dejando solamente el cobre con el dibujo de las pistas deseadas.

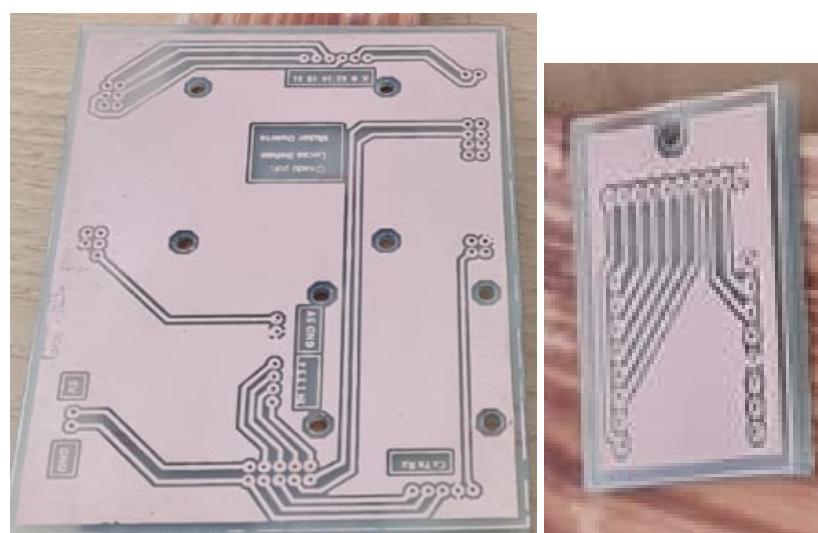


Figura 22: Impresión del circuito post acido y barnizadas.



Centro de Técnicas Analógico-Digitales (CETAD)

A continuación podemos ver la placa finalizada a la cual se le soldaron los componentes deseados y por último se obtuvo el “Poncho” para la EDU-CIAA correspondiente.

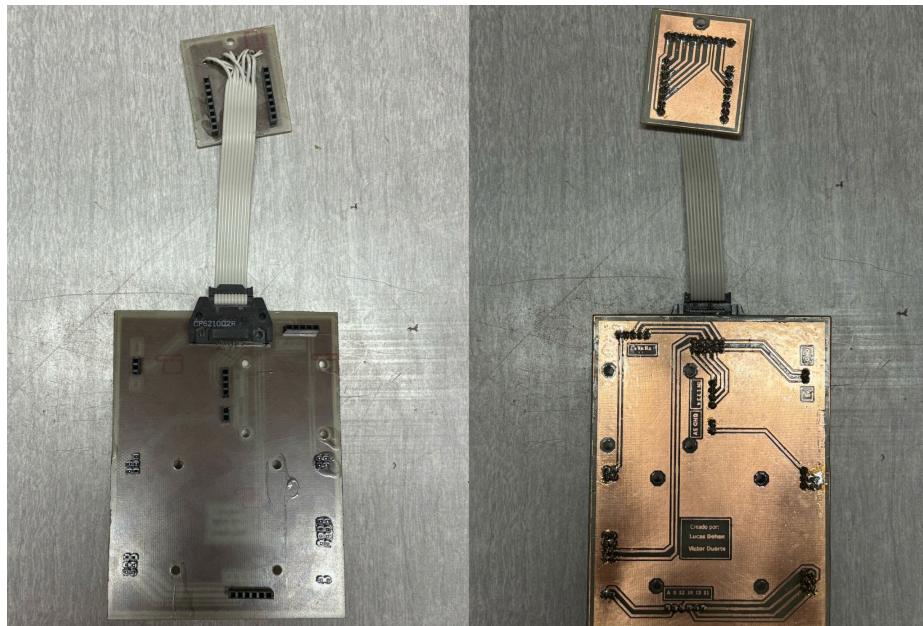


Figura 23: “Poncho” finalizado.

Diseño del software

El diseño del software del sistema abarca distintos módulos encargados de la comunicación y control del RME. Cada uno de estos módulos cumple una función específica para garantizar el correcto funcionamiento del robot:

Programación en la EDU-CIAA: Controla los motores y la lógica de movimiento del RME.

Programación en el ESP32-CAM: El ESP32-CAM se encarga de gestionar la captura y transmisión de imágenes, la comunicación con la API móvil, así como el proceso de barrido para la obtención de medidas



Centro de Técnicas Analógico-Digitales (CETAD)

API móvil: Permite la interacción con el usuario, facilitando el control remoto del robot.

Servidor FTP: Utilizado para el almacenamiento y gestión de imágenes, alojado en la infraestructura de la facultad.

A continuación, se detalla la implementación de cada uno de estos módulos.

EDU-CIAA

Tabla 3. Módulos desarrollados para la EDU-CIAA

Módulo	Descripción
ctrl_marchas	Controla el movimiento de los motores de corriente continua
esp32_cam	Administra la comunicación con la ESP32-CAM
main	Programa principal
mensajes	Procesa los mensajes recibidos por UART.

Excepto el programa principal (main), todos los módulos cuentan con un archivo fuente y un archivo de encabezado. El main utiliza una arquitectura superloop.

Además, se implementó un módulo denominado “moduloint”, cuyo propósito es reducir el consumo de energía en la EDU-CIAA mediante la activación de un modo sleep, del cual debería despertarse a través de interrupciones. Sin embargo, si bien el módulo funciona, se ha observado que la EDU-CIAA entra en modo sleep con el comando “__WFI();”, pero se despierta de inmediato por razones aún desconocidas. Por este motivo, no se detallará su funcionamiento en este informe.



Centro de Técnicas Analógico-Digitales (CETAD)

ctrl_marchas

En este módulo de software, se han definido los movimientos del RME: adelante, atrás, giro izquierda y giro derecha. Para los desplazamientos lineales hacia adelante y hacia atrás, se optó por utilizar PWM para controlar la velocidad. Para realizar los giros, se implementó un sistema de tracción diferencial, dado que las ruedas del RME son fijas y no poseen un mecanismo de dirección. El giro se logra variando la velocidad de las ruedas de cada lado: las ruedas del lado opuesto al giro se mueven a mayor velocidad, mientras que las ruedas del lado hacia donde se desea girar se mueven a menor velocidad o incluso se detienen. De esta forma, el diferencial de velocidad entre ambos lados genera un radio de giro, permitiendo que el robot cambie de dirección.

Métodos públicos:

- void `ctrl_Init (void)`: Configura los pines GPIO conectados a los motores como salida y se inicializan las señales PWM correspondientes.
- void `ctrl_ruedas (int pos_x, int pos_y)`: Controla el movimiento de las ruedas en función de los parámetros de entrada.
 - **pos_x (velocidad)**: Puede tomar valores en el rango de [-1,1], donde valores positivos representan avance y valores negativos representan retroceso.
 - **pos_y (giro)**: Puede tomar valores en el rango de [-1, 1], donde valores negativos indican giro a la izquierda, valores positivos indican giro a la derecha, y 0 representa un movimiento recto. El valor absoluto determina la intensidad del giro.

Los valores `pos_x` y `pos_y` son utilizados en los métodos privados (`obtener_vel` y `ctrl_motor`) para calcular la velocidad de cada rueda y controlar su alimentación.

Métodos privados:

- int `obtener_vel (int pos_x, int pos_y, pwmMap_t pwm)`: Este método calcula la velocidad final de un motor en función de su posición (motor A: izquierda o motor B: derecha)



Centro de Técnicas Analógico-Digitales (CETAD)

1. Determina si el motor corresponde al lado derecho o izquierdo haciendo uso del pwm.
2. En base a pos_y, verifica si el motor está en el lado hacia el cual se desea girar.
 - a. Si el motor está en el lado del giro, su velocidad se reduce proporcionalmente al valor absoluto de pos_y.
 - b. Si el motor está en el lado opuesto al giro, mantiene la velocidad establecida por pos_x.

La velocidad resultante, se utilizará luego en el método “ctrl_motor”.

- void ctrl_motor (int sentido, int velocidad, pwmMap_t PWM, gpioMap_t positivo, gpioMap_t negativo): Este método controla la activación de los motores mediante el puente H, en función de los parámetros recibidos:
 1. **sentido**: Define la dirección de giro del motor (avance o retroceso).
 2. **velocidad**: Establece la intensidad de la señal PWM para regular la velocidad del motor.
 3. **PWM**: Especifica el canal PWM utilizado para modular la velocidad.
 4. **positivo y negativo**: Representan los pines GPIO conectados al puente H, determinando la polaridad de la alimentación del motor.

esp32_cam

Métodos públicos:

- void esp32_init(void): Inicializa la comunicación UART con el ESP32-CAM del lado de la EDU-CIAA.
- void receiveMessage(char *cadena): Copia lo recibido en el buffer de la UART en cadena.

Métodos privados:

- void esp32CleanRxBuffer(void): Limpia el buffer de la UART.



Centro de Técnicas Analógico-Digitales (CETAD)

mensajes

Métodos públicos:

- void procesarMensaje(char *cadena): Procesa el mensaje recibido desde la ESP32-CAM para identificar qué tipo de mensaje es y si es válido. Las clases de mensajes válidos pueden ser:
 - 200: Identifica que se establece una comunicación entre la EDU-CIAA y la ESP32-CAM.
 - 400: Identifica que se finaliza la comunicación.
 - Coordenadas (pos_x, pos_y): Utilizadas para el desplazamiento del RME.

Los mensajes no válidos se descartan.

main

El programa principal consta de dos partes fundamentales: el **setup**, que inicializa la placa y los módulos del sistema, y el **loop**, que, en intervalos regulares, revisa el buffer del ESP32-CAM y realiza el procesamiento correspondiente de los datos.

Pseudocódigo del setup:

Se inicializa la placa EDU-CIAA
Se inicializa el puente H
Se inicializa la conexión UART con la ESP32-CAM
Se inicializa el módulo de interrupciones.
Se hace un retardo de 100 ms

Pseudocódigo del loop:

Mientras que haya datos que leer en la UART
Se recibe el mensaje
Se procesa el mensaje
Se limpia el buffer



Centro de Técnicas Analógico-Digitales (CETAD)

ESP32-CAM

Tabla 4. Módulos desarrollados para el ESP32-CAM

Módulo	Descripción
conexion_udp	Administra y procesa las comunicaciones UDP
cámara	Administra la cámara de la ESP32-CAM
pins_camara	Archivo de configuración de pines para el uso de la cámara
tf_lc02	Controla el sensor LiDAR
motor_28byj48	Controla el motor paso a paso
ftp	Administra la conexión con el servidor FTP
esp32_servidor_ciaa	Programa principal

conexion_udp

Este módulo de software gestiona la comunicación UDP entre el ESP32-CAM y la aplicación móvil. Los paquetes UDP pueden recibirse a través del puerto de control, cuando corresponden a mensajes de inicio o finalización de la comunicación, o mediante el puerto de datos, cuando contienen comandos enviados por el usuario.

Dado que UDP es un protocolo sin conexión, se incorporó un número de secuencia en los paquetes para verificar su orden de llegada. En caso de recibir paquetes desactualizados o fuera de orden, estos son descartados.



Centro de Técnicas Analógico-Digitales (CETAD)

Métodos públicos:

- void setup_udp(IPAddress ip): Inicia el servidor UDP.
- bool getComunicacion(): Retorna true si existe una comunicación establecida, false en caso contrario.
- bool hayPaqueteDatos(char *buffer, size_t bufferSize): Retorna true si existe un paquete en el buffer de datos.
- bool hayPaqueteControl(char *buffer, size_t bufferSize): Retorna true si existe un paquete en el buffer de control.
- void conectarCliente(): Establece la comunicación con el usuario.
- void desconectarCliente(): Finaliza comunicación con el usuario
- void limpiarPaquetes (char *buffer, size_t bufferSize): Limpia el buffer para descartar paquetes.
- void pausarComunicacion(): Cierra el puerto de datos y vacía el buffer para que el sistema no reciba mensajes en el puerto de datos.
- void reanudarComunicacion(): Abre el puerto de datos para reanudar la comunicación con el usuario.

Métodos privados:

- void abrirPort(int port): Abre el puerto declarado en el parámetro port.
- void cerrarPort(int port): Cierra el puerto declarado en el parámetro port.
- void enviarMensaje(int port, IPAddress ip, const char* mensaje): Envía el mensaje al usuario conectado.
- bool getPortControlActivo(): Retorna true si el puerto de control está activo, false en caso contrario.
- bool getPortDatosActivo(): Retorna true si el puerto de datos está activo, false en caso contrario.
- void descartarPacket(int packetSize,char *buffer, size_t bufferSize): Descarta paquetes de datos.



Centro de Técnicas Analógico-Digitales (CETAD)

- bool paqueteEnOrden(char *buffer, size_t bufferSize): Retorna true si el paquete es más reciente al último que fue procesado en base al número de secuencia. False en caso contrario.

camara

Métodos públicos:

- void setup_camara(): Inicia la camara
- size_t tomarFoto(uint32_t colores[cantMedidas], uint8_t **bmp_buf): Captura una imagen y la procesa para enriquecerla con los datos obtenidos del barrido.
- void alternarModo(): Alterna la cámara entre el modo de transmisión en vivo y captura de fotos
- void startCameraServer(): Inicia un servidor para la transmisión de la cámara.

Métodos privados:

- bool capturar_foto(camera_fb_t **fb): Toma una foto y la guarda en el buffer de la cámara.
- void dibujarLinea(uint8_t * bmp_buf, uint32_t colores[cantMedidas]): Recorre el buffer de imagen y traza sobre ella las marcas para enriquecer la imagen.
- modificarPixelBMP(uint8_t * bmpData, int x, uint32_t color): Modifica los canales del pixel en base al color y posición pasados por parámetro.
- esp_err_t stream_handler(httpd_req_t *req): Administra la transmisión de la cámara



Centro de Técnicas Analógico-Digitales (CETAD)

pins_camara

Este encabezado contiene las definiciones de los pines del ESP32 utilizados por la cámara.

Para este proyecto, se empleó el modelo AI-Thinker ESP32-CAM.

motor_28byj48

Este módulo de software gestiona el control del motor paso a paso en el cual se monta el sensor de proximidad LiDAR. Su función principal es regular el movimiento del motor para realizar el barrido necesario en la captura de datos.

Para determinar el giro requerido, se realizó un análisis del ángulo de barrido necesario para cubrir el área correspondiente a la imagen capturada por la cámara. En la **Figura 24** se muestra la relación entre el ancho de la imagen (representado por la línea roja) y la trayectoria del LiDAR (representada por la media circunferencia).

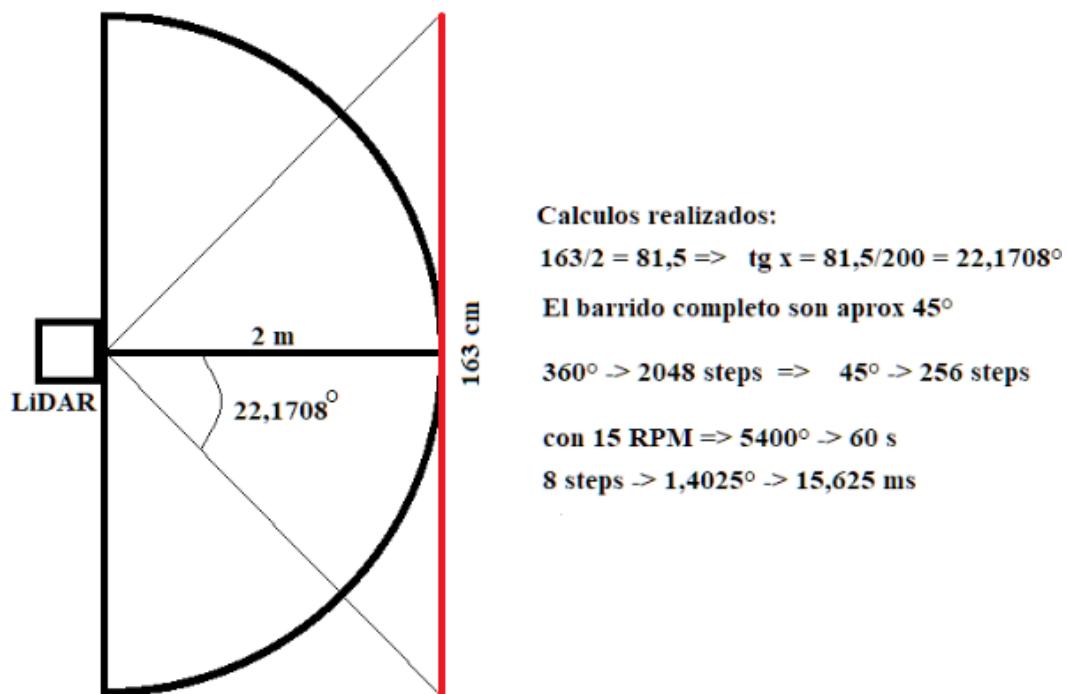


Figura 24: Cálculo de superficie medida con LiDAR.



Centro de Técnicas Analógico-Digitales (CETAD)

Como se observa en la imagen, los cálculos indican que para cubrir un barrido de **45°** es necesario realizar **256 pasos (steps)** con el motor. A la máxima velocidad operativa del motor (**15 RPM**), este movimiento se completa en **0,5 segundos**.

Además, se calculó la distancia máxima a la que debe situarse el RME para garantizar que el sensor LiDAR pueda alcanzar todas las mediciones dentro del área capturada por la cámara. Como se muestra en la **figura 25**, si el RME se posiciona de manera que los extremos de la imagen estén a 2 metros del sensor LiDAR, la distancia lineal al primer objeto debe ser de **185,21 cm**.

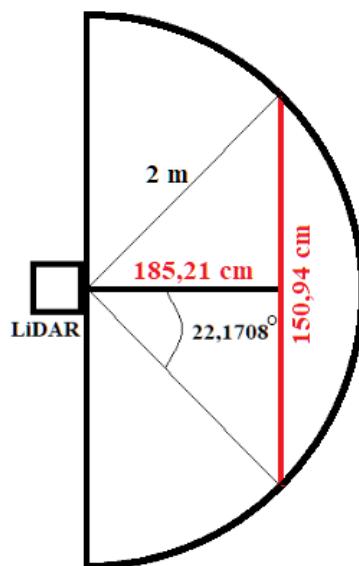


Figura 25: Cálculo de la distancia máxima del LiDAR.



Centro de Técnicas Analógico-Digitales (CETAD)

Métodos públicos:

- void setup_motor(int rpm): Inicia el motor paso a paso, configurando el RPM de trabajo.
- void moverMotor(int pasos, int dirección): Mueve el motor en una cantidad de pasos. La dirección determina si gira a la derecha o a la izquierda.

tf_lc02

El sensor **TF LC-02** realiza cada medición en **33 ms**. Para lograr un barrido eficiente, equilibrando tiempo y precisión, se tuvo en cuenta que el movimiento del motor es bloqueante, lo que significa que debe detenerse cada vez que se realice una medición.

Basándose en los cálculos previos del módulo **motor_28byj48**, se determinó que la mejor estrategia para minimizar el tiempo total del barrido sin modificar el hardware es reducir la cantidad de mediciones manteniendo una cobertura adecuada de la superficie escaneada.

Para ello, se estableció que, en un barrido de **45°** (256 pasos), la opción más eficiente es realizar **16 mediciones**, distribuidas uniformemente cada **16 pasos**. Con esta configuración, el tiempo total requerido para completar una medición es:

$$\text{Tiempo Mediciones} + \text{Tiempo 256 steps} = \text{Tiempo Total}$$

$$(16 \text{ mediciones} \times 33 \text{ ms}) + 0.5 \text{ s} = 1.028 \text{ s}$$

De este modo, se consigue un barrido preciso en un tiempo menor a **1.5 segundos**, optimizando el rendimiento del sistema.

Métodos públicos:

- void setup_lidar(): Inicia el sensor LiDAR
- int getDistancia(): Retorna la distancia medida por el sensor como tal en milímetros
- int getDistanciaCalibrada(): Retorna la distancia medida por el sensor compensando el desfase entre la cámara y el sensor en milímetros.



Centro de Técnicas Analógico-Digitales (CETAD)

Métodos privados:

- `getLidarData(TF* lidar)`: Procesa los bytes recibidos por el serial desde el LiDAR y los almacena.

ftp

Métodos públicos:

- `void setup_ftp()`: Realiza la configuración para la comunicación por protocolo FTP
- `void subirFoto(uint8_t * buf, size_t buf_len)`: Sube la foto al servidor FTP.
- `void conectarWiFi()`: Realiza la conexión a WiFi

Métodos privados:

- `String getTimestamp()`: Genera un timestamp o marca de fecha y hora como nombre de archivo.

esp32_servidor_ciaa

Este módulo es el programa principal del ESP32-CAM. Al igual que el programa principal de la EDU-CIAA, consta de dos partes fundamentales: el **setup**, que inicializa los módulos del sistema, y el **loop** que recibe los mensajes de la aplicación y los procesa.

El loop del sistema se describe en pseudocódigo a continuación:

Mientras el sistema esté en ejecución:

 Si hay al menos un dispositivo conectado a la red WiFi:

 Mientras exista comunicación con un cliente:

 Si hay un paquete de datos en el buffer:

 Convertir el contenido del buffer en una cadena de texto.

 Procesar el mensaje recibido.

 Actualizar el tiempo del último mensaje recibido.

 Sino:

 Si el tiempo transcurrido desde el último mensaje recibido supera el tiempo de espera (timeout):

 Desconectar el cliente de la CIAA por timeout.

 Finalizar la comunicación con el cliente, liberando el canal para nuevas



Centro de Técnicas Analógico-Digitales (CETAD)

conexiones.

Si hay un paquete de control en el buffer:

Convertir el contenido del buffer en una cadena de texto.

Procesar el mensaje recibido para verificar si es una solicitud de inicio de comunicación.

Sino:

Entrar en estado de espera hasta que un dispositivo se conecte a la red.

Este módulo además ejecuta el algoritmo para el realizado del barrido y enriquecimiento de imágenes acorde al algoritmo descrito en pseudocódigo a continuación:

Si no tiene conexión WiFi, intentar conectarse.

para ($i = 0..15$) {

Determinar la índice acorde a la dirección de barrido

Guardo el color que corresponde a la distancia medida por el sensor en un vector en el índice determinado;

Giro en la dirección correspondiente 16 pasos;

}

Guardo el color de la última distancia medida en la posición final del vector acorde a la dirección de barrido;

Alternar la variable de posición de sensor;

Si la toma de imagen y enriquecerla con medidas fue exitoso {

Esperar 200 ms para prevenir un pico de consumo;

Subir imagen al servidor;

}

Liberar memoria del buffer;

El sistema verifica inicialmente la disponibilidad de conexión WiFi para la posterior subida al servidor FTP. En caso de no estar conectado, intenta establecer la conexión. A continuación, se inicia un proceso iterativo de 16 ciclos en el cual se determina un índice para almacenar en un vector en función de la dirección de barrido del sensor. En cada iteración, se mide la distancia con el sensor LiDAR y se guarda el color correspondiente en un vector en la posición determinada por el índice calculado previamente. Simultáneamente, se realiza un giro con el motor PaP en la dirección especificada, avanzando 16 pasos en cada iteración.



Centro de Técnicas Analógico-Digitales (CETAD)

Al finalizar el ciclo, se almacena el color correspondiente a la última medición en la posición final del vector, asegurando la coherencia con la dirección de barrido establecida. Posteriormente, se alterna la variable que controla la posición del sensor para modificar su orientación en la siguiente ejecución.

Para enriquecer las imágenes se propuso mapear las distancias medidas a colores en formato RGB, este formato asigna un valor de 1 byte para cada color primario. Como el sensor LiDAR devuelve la distancia medida en milímetros, para que los valores de cada color puedan ser representados correctamente se convierte la medida en centímetros. Para realizar el enriquecimiento, se trazan marcas sobre la imagen tomada con las medidas traducidas a colores RGB. Estas marcas se trazan cada 50 píxeles con el color correspondiente en la ubicación en la que se tomó la medida.

Métodos:

- `bool conexion_ciaa()`: Inicia comunicación con la CIAA desde el lado de la ESP32-CAM.
- `void procesar_mensaje(String mensaje)`: Procesa el mensaje recibido de la aplicación. De acuerdo al mensaje, se realizará el siguiente procesamiento:
 - 200: Se conecta a la EDU-CIAA y establece la comunicación UDP con el usuario para futuros mensajes. Se coloca el sensor en posición para realizar barridos.
 - 400: Informar a la EDU-CIAA que la comunicación finaliza, finalizar comunicación con el usuario y centrar el sensor.
 - foto: Se pausa la transmisión de la cámara y pausa la comunicación UDP para realizar el barrido. Al finalizar, se reanuda la transmisión y comunicación.
 - Coordenadas: Se envían las coordenadas a la EDU-CIAA.
- `void barrido()`: Realiza el barrido acorde al algoritmo descrito anteriormente.
- `uint32_t mapearColor(int distancia)`: Mapea la distancia en milímetros a un color en formato RGB. Para armar el color, se toma la distancia en centímetros y se guarda su



Centro de Técnicas Analógico-Digitales (CETAD)

valor en el canal del color G (verde), con el objetivo de que al momento de entrenar una IA se utilice el valor que fue medido. Si el sensor no detecta ningún objeto en su rango de trabajo, devuelve 8888 mm por lo que durante el mapeado se traduce dicho valor a 255. Para apreciar mejor a ojo la distancia medida, se implementa un gradiente rojo-verde en donde los tonos rojos indican objetos detectados a corta distancia y con verde a objetos detectados a mayor distancia o si no se detectó nada.

Snaprunner

Es una aplicación móvil desarrollada en Flutter, actualmente programada para Android. Gracias a su implementación en Flutter, el mismo código puede utilizarse también en iOS y Windows sin modificaciones significativas.

La aplicación se compone de las siguientes vistas:

- **Home_Page**
- **control_page**

Las cuales hacen uso de los siguientes servicios para funcionar:

- **wifi_service**
- **udp_service**

home_page

La vista de inicio es la pantalla principal de la aplicación y permite al usuario establecer la conexión con el **ESP32-CAM**.

Inicialmente, se muestra un botón denominado "**Conectar**", el cual permite al usuario conectarse a la red WiFi del **ESP32-CAM**.

Si la conexión se establece con éxito, la interfaz actualizará su estado y mostrará dos nuevos botones:



Centro de Técnicas Analógico-Digitales (CETAD)

- "**Controles**": Redirige al usuario a la vista de control (**control_page**), donde podrá operar el vehículo y visualizar la transmisión de la cámara.
- "**Desconectar**": Finaliza la conexión con el **ESP32-CAM**, restableciendo la vista a su estado inicial y volviendo a mostrar únicamente el botón "**Conectar**".

control_page

La vista de control es la interfaz desde la cual el usuario puede operar el vehículo y visualizar la transmisión en tiempo real de la cámara. Está compuesta por tres elementos principales:

- Joystick: Permite controlar el movimiento del RME mediante la captura de coordenadas en los ejes X e Y. Haciendo uso de la librería: **flutter_joystick**
- **Pantalla de cámara**: Muestra en tiempo real la transmisión de la cámara integrada en el **RME**, permitiendo al usuario visualizar el entorno.
- **Botón de captura**: Permite al usuario capturar una imagen de la visualización actual de la cámara del **RME**.

wifi_service

El servicio contiene la lógica necesaria para gestionar la conexión y desconexión de la red WiFi. Su función principal es establecer la comunicación entre la aplicación y el **ESP32-CAM**, permitiendo una conexión estable y segura. Además, incluye los mecanismos para finalizar la conexión cuando el usuario lo requiera.

Métodos:

Future<bool> connectToWiFi() async: Este método establece la conexión WiFi con la red especificada mediante su **SSID** y clave de acceso. La conexión se realiza de forma segura utilizando el protocolo **WPA**, garantizando la autenticidad y protección de los datos. Devuelve un **booleano** que indica si la conexión fue exitosa (**true**) o fallida (**false**).



Centro de Técnicas Analógico-Digitales (CETAD)

Future<void> requestPermissions() async: Este método solicita al usuario los permisos necesarios para establecer la conexión WiFi en dispositivos Android. Dado que las versiones recientes de Android requieren permisos adicionales para el escaneo y conexión a redes WiFi, la función gestiona solicitudes como **acceso a la ubicación** y otros permisos relevantes. Devuelve un **booleano** indicando si los permisos fueron otorgados (**true**) o denegados (**false**).

Future<bool> disconnectFromWiFi() async: Este método se encarga de **desconectar** el dispositivo de la red WiFi actualmente establecida con el ESP32. Garantiza que la conexión sea finalizada correctamente y restablece el estado de la aplicación para permitir una nueva conexión en el futuro. Devuelve un **booleano**, donde **true** indica que la desconexión fue exitosa y **false** en caso de error.

udp_service

El servicio **UDP Service** gestiona el **envío y recepción de datos** a través de la red WiFi establecida con el ESP32. Implementa la lógica necesaria para la comunicación mediante el **protocolo UDP**, permitiendo una transmisión de datos rápida y eficiente entre la aplicación y el **RME**. Para conexión utiliza el puerto de datos, el cual le permite al ESP32, enviarle el puerto de control al cual se tiene que conectar para enviar información de control.

Métodos:

Future<bool> connectToServer() async: Se encarga de escuchar el puerto de datos hasta recibir el puerto de control, momento en el cual establece la conexión.

void cerrarComunicacion(): Se encarga de cerrar los puertos de envío y recepción, finalizando así la comunicación.



Centro de Técnicas Analógico-Digitales (CETAD)

void enviarDatos(String msj) async: Verifica la estabilidad de la conexión de red. Si la conexión es estable, crea el paquete de datos a enviar y lo transmite a través del puerto de control.

Ensayos y mediciones

1. Conexión de la EDU-CIAA al puente H y control de las ruedas

Se conectó la placa EDU-CIAA al puente H y se cargó un programa que permite controlar el movimiento de las dos ruedas iniciales del vehículo. Este paso inicial fue fundamental para garantizar el funcionamiento básico del sistema de movilidad.

Se realizaron dos pruebas: una en la que las ruedas giraban a máxima velocidad, y posteriormente, se implementó una mejora en la que las ruedas se regulan mediante señales PWM, permitiendo un control más preciso de su velocidad. Para concluir este ensayo, dado que las ruedas requieren una fuerza mínima para superar la resistencia del rozamiento estático, se determinó que el valor mínimo de la señal PWM necesario para que las ruedas comenzaran a moverse fue del 38% de ciclo de trabajo.

“Link al video que muestra su funcionamiento inicial”

2. Integración del ESP32-CAM con la EDU-CIAA

Una vez comprobada la correcta operación de las ruedas, se procedió a conectar el módulo ESP32-CAM a la EDU-CIAA. Se estableció una comunicación entre ambos dispositivos, lo que permitió que el ESP32-CAM pudiera interactuar con la placa para recibir y enviar datos de control.

Se realizó una prueba exitosa en la que se desarrolló un código para el ESP32-CAM, permitiéndole enviar información de coordenadas a la EDU-CIAA, para que esta pudiera mover el RME de acuerdo con las indicaciones recibidas.



Centro de Técnicas Analógico-Digitales (CETAD)

3. Creación de la red Wi-Fi y desarrollo de la aplicación móvil

Una vez establecida la comunicación, se configuró una red Wi-Fi en el ESP32-CAM. Para probar la comunicación UDP, se realizó una prueba exitosa mediante un script en el cual se enviaban comandos al ESP32 que simulaban coordenadas. Estos comandos eran enviados al ESP32, quien los transmitía a la EDU-CIAA para activar los motores. Los mensajes enviados por UDP fueron los siguientes:

- "200" (para iniciar la comunicación)
- "1.0, 0.0"
- "0.0, 0.0"
- "-1.0, 0.0"
- "0.0, 0.0"
- "1.0, 0.5"
- "0.75, -0.5"
- "0.0, 0.0"
- "400" (para finalizar la comunicación)

Este proceso permitió el desarrollo de una aplicación móvil para controlar el vehículo de manera remota. La aplicación proporcionó la interfaz necesaria para gestionar los movimientos del RME.

“Link al video que muestra su funcionamiento con la aplicación” 4. Configuración de la cámara ESP32-CAM

Al lograr con éxito el control remoto del vehículo, se procedió a configurar la cámara del ESP32-CAM. Esto permitió transmitir imágenes en vivo y capturar fotos cuando fuera necesario, agregando una funcionalidad de monitoreo visual al sistema.

En la aplicación, dentro de la vista **control_page**, se incorporaron dos joysticks: el izquierdo, encargado de controlar la velocidad, y el derecho, para el giro. Además, se incluyó una visualización en tiempo real de los valores obtenidos por cada joystick, permitiendo



Centro de Técnicas Analógico-Digitales (CETAD)

monitorear su funcionamiento. A esta vista se le añadió la transmisión en vivo de la cámara, ubicada en el centro de la interfaz, así como un botón en la esquina inferior derecha para capturar imágenes, como se muestra en la siguiente figura.

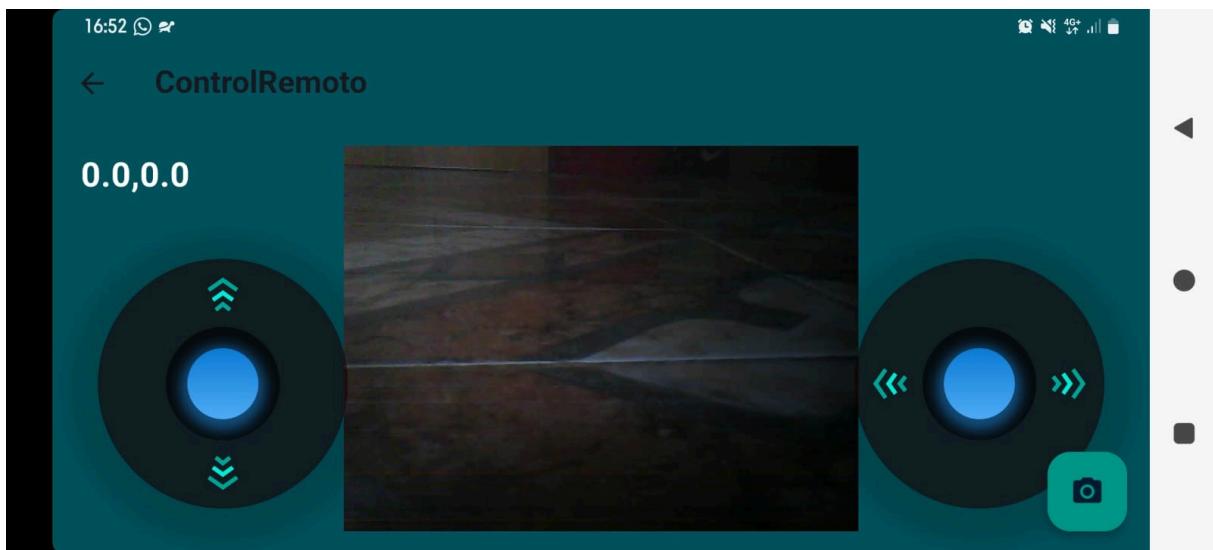


Figura 26: Captura de la aplicación “Vista control_page”.

5. Ensamblaje y desarrollo de los módulos controlador paso a paso y sensor LiDAR

Tras la finalización de los ensayos previos, se integró un controlador ULN2003A al sistema con el propósito de gestionar el motor paso a paso (PAP) desde el módulo ESP32-CAM. Para ello, se implementó en el código del ESP32-CAM el módulo correspondiente, permitiendo la ejecución de un movimiento angular de 45°, con el objetivo de abarcar el ancho de la imagen capturada por la cámara.

Adicionalmente, se incorporó un sensor LIDAR montado sobre el motor PAP, lo que posibilitó la adquisición de datos de proximidad en tiempo real. Con esta configuración, se desarrolló un algoritmo que permite la sincronización del barrido del sensor LIDAR con la



Centro de Técnicas Analógico-Digitales (CETAD)

captura de imágenes, optimizando así la exploración del área de interés y mejorando la precisión del sistema en la obtención de datos espaciales.

“Link al video que muestra el barrido del sensor LiDAR”

6. Conexión con el servidor FTP para el envío de imágenes enriquecidas

Finalmente, se estableció una conexión con un servidor FTP para la transmisión de imágenes capturadas por la cámara del ESP32-CAM enriquecidas con los datos obtenidos del barrido. Preliminarmente, se realizaron pruebas de almacenamiento de imágenes configurando una computadora personal como servidor FTP utilizando el software “Rebex Tiny FTP Server”. Este programa permite levantar un servidor FTP especializado para pruebas, lo cual cumplía con los requisitos del proyecto.

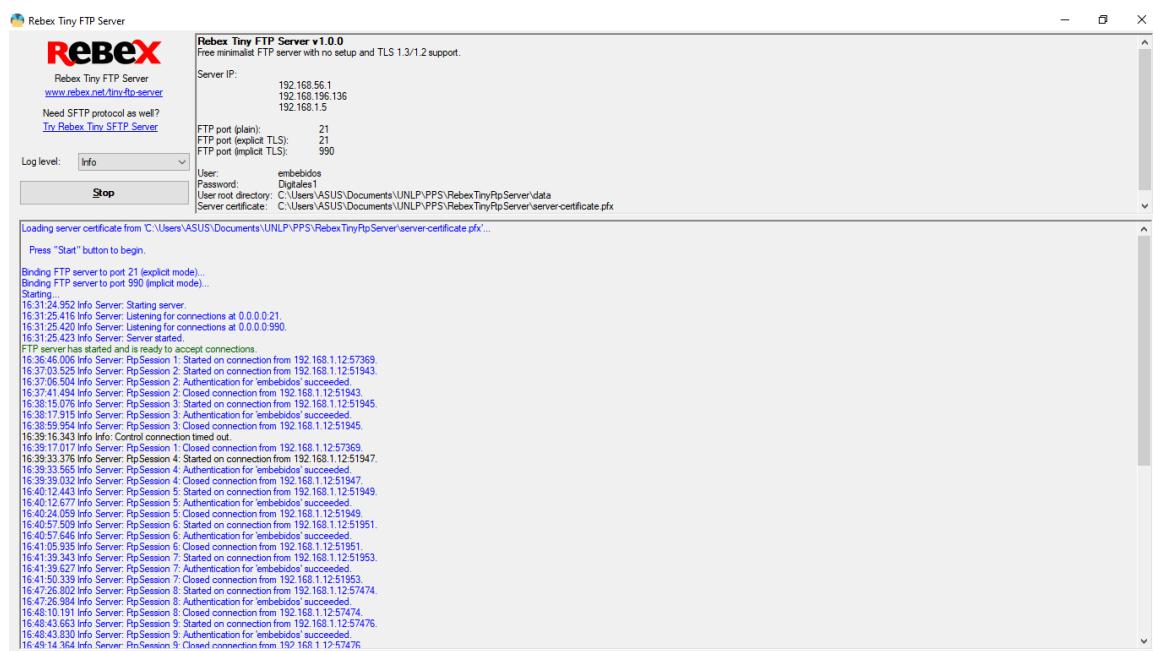


Figura 27: Ensayo de uso de Rebex Tiny FTP Server.

En la Figura 27 se muestra el servidor en funcionamiento. Al iniciarse, el servidor abre los puertos 21 y 990, dependiendo de si se utiliza TLS (protocolo de seguridad) de forma explícita o no.



Centro de Técnicas Analógico-Digitales (CETAD)

Con este servidor de pruebas, se calculó la velocidad de transferencia en Mbps. Se realizaron varios barridos y se calculó el promedio en base a los logs del servidor. La fórmula utilizada fue:

$$\text{Velocidad de transferencia (Mbps)} = \frac{\text{Tamaño de archivo (Bits)}}{\text{Tiempo de transferencia (segundos)}} * \frac{1}{10^6}$$

El $\frac{1}{10^6}$ en la formula es para convertir de bits a Megabits. Los cálculos nos dieron un promedio de 2 Mbps.

Dificultades en proceso de diseño

Durante el ensamblaje del hardware

Inicialmente, el vehículo presentaba dificultades de movilidad debido al uso de dos ruedas motrices junto con una rueda de giro libre. Esta configuración no permitía un desplazamiento adecuado, por lo que se optó por incorporar cuatro ruedas, mejorando así el agarre sobre la superficie y la estabilidad. Este cambio se detalla en la sección **diseño de hardware**.

Por otra parte, a medida que el proyecto avanzaba y se incorporaban nuevos sensores, el espacio físico disponible resultaba limitado. Esto requirió varias modificaciones en el diseño para reorganizar la disposición de los componentes. Cada cambio de ubicación implicó la necesidad de realizar nuevas pruebas para verificar el correcto funcionamiento de los elementos. Esto añadió complejidad al proceso de ensamblado, ya que algunos cableados se dañaron durante las modificaciones, mientras que otros dejaban de ser compatibles, lo que requería ajustes adicionales.



Centro de Técnicas Analógico-Digitales (CETAD)

Implementación de interrupciones

Durante la implementación de interrupciones para reducir el consumo de energía en las placas EDU-CIAA y ESP32-CAM, se identificaron diversas dificultades en ambas. En el caso de la EDU-CIAA, como se menciona en la sección de software, al ingresar en modo sleep, la placa se activa de inmediato por razones no determinadas, lo que impide el ahorro de energía esperado.

Por otro lado, al intentar aplicar una estrategia similar en la ESP32-CAM, se observó que todos los modos de suspensión disponibles desactivan la antena WiFi. Como consecuencia, la aplicación no puede establecer comunicación con el RME. Dado que la conectividad es un aspecto crítico para el funcionamiento del sistema, se decidió finalmente descartar esta funcionalidad en la ESP32-CAM.

Desarrollo de la aplicación móvil

Fue necesario estudiar un nuevo lenguaje de programación, Dart, junto con el framework Flutter, para diseñar y desarrollar la aplicación móvil destinada a dispositivos Android, asegurando su compatibilidad y funcionalidad acorde a los requerimientos del proyecto.

Incompatibilidad de pines en el ESP32-CAM

Se intentó utilizar el slot de memoria SD del ESP32-CAM para almacenar temporalmente las imágenes capturadas cuando el RME no tuviera conexión con el servidor FTP, permitiendo así su posterior carga una vez restablecida la conexión. Sin embargo, esta implementación no fue posible, ya que los pines requeridos por el ESP32-CAM para la tarjeta SD también estaban siendo utilizados para otra conexión dentro del proyecto, generando un conflicto de hardware.



Centro de Técnicas Analógico-Digitales (CETAD)

Cambios de protocolo de transferencia de archivos

En un principio, se planeó realizar la transmisión de los datos mediante el protocolo MQTT, aprovechando sus beneficios para permitir una comunicación eficiente con recursos mínimos. En la arquitectura MQTT, existen dos tipos de sistemas: clientes y brókeres. Un bróker actúa como servidor intermediario entre los clientes, recibiendo y enviando comunicaciones. Los clientes, en lugar de comunicarse directamente, se suscriben a canales a través de los cuales pueden enviar o recibir datos. Cada cliente puede estar suscrito a uno o varios canales simultáneamente. Para que este modelo funcione, es necesario contar con un bróker que actúe como intermediario, lo que agregaría un servicio adicional al proyecto. Sin embargo, dado que el sistema solo admite como máximo una conexión desde la aplicación (un máximo de dos conexiones al bróker: aplicación y servidor), el uso de un bróker no proporciona ventajas significativas frente a una conexión directa. Además, aunque MQTT puede transmitir archivos de imagen, no está diseñado para manejar grandes volúmenes de datos. Fue concebido para transmisiones de menor tráfico, lo que limita su eficiencia en este contexto. Por estas razones, se descartó el uso de MQTT y se optó por FTP para la transmisión de datos.



Centro de Técnicas Analógico-Digitales (CETAD)

Conclusiones

En los objetivos del proyecto se enumeraron varias metas para nuestro prototipo.

Tabla 5. Grado de cumplimiento de los objetivos solicitados.

Objetivo definido	Grado de cumplimiento del objetivo
Control de motores - Algoritmos de navegación EDU CIAA-ESP32.	Se cumplió con el objetivo.
Aplicación web/móvil - Comunicación WiFi.	Se logró desarrollar la aplicación móvil, aunque actualmente solo está disponible para dispositivos Android, con la posibilidad de futuras expansiones.
Captura de Imágenes con el ESP32-CAM	Se logró configurar el ESP32-CAM para procesar los mensajes que provienen de la aplicación móvil y la cámara integrada para capturar imágenes desde el RME.
Sincronización de Barrido: LiDAR y motor paso a paso	Se logró implementar un barrido sincronizado con las medidas del LiDAR y el movimiento del motor paso a paso
Implementación de transmisión WiFi - Sincronización de datos con servidor FTP	Se logró implementar una comunicación para transmisión de las imágenes enriquecidas por el barrido al servidor FTP.
Diseño de PCB - Alimentación y protección.	Se logró el desarrollo de PCB “poncho” tanto para la EDU-CIAA como para el ESP32-CAM.
Integración y Pruebas	Todos los sistemas fueron integrados con éxito en el RME, cumpliendo con todas las tareas solicitadas.

Según la tabla presentada, podemos concluir que nuestro proyecto ha cumplido satisfactoriamente con todas las tareas establecidas en los objetivos propuestos. El producto final es capaz de navegar de manera eficiente comandado de forma inalámbrica, capturando imágenes del entorno y enviándolas al servidor FTP, tal como se solicitó.



Centro de Técnicas Analógico-Digitales (CETAD)

Tiempo dedicado y Presupuesto final

Con el fin de demostrar las horas invertidas en el proyecto, se realizó una tabla con las tareas y horas dedicadas para cada una.

Tabla 1. Actividades realizadas por los integrantes.

Tarea	Horas dedicadas a la tarea
Ensamble de hardware: Se realizó el montaje de los componentes en el chasis del sistema, incluyendo la placa EDU-CIAA, el puente H, los motores, las ruedas, fuente stepdown y las baterías.	5
Desarrollo del software de navegación	4
Desarrollo del software de comunicación ESP32 con Edu ciaa	8
Desarrollo de la aplicación móvil en Flutter	42
Desarrollo módulo de comunicación Access Point de ESP32-CAM	20
Configuración de la cámara del ESP32-CAM para capturar imágenes y video	12
Ampliación de aplicación móvil: Inclusión de video de la ESP32-CAM y botón para captura de imágenes.	7
Ampliación del ensamblaje de hardware: Se incorporó al chasis un motor paso a paso con su respectivo controlador, soporte y fijación del sensor LiDAR	5
Mediciones de área de barrido y desarrollo de software controlador motor PaP	11
Desarrollo de software sensor LiDAR	8
Desarrollo de algoritmo de barrido	20



Centro de Técnicas Analógico-Digitales (CETAD)

sincronizado y enriquecimiento de imagen	
Desarrollo y pruebas de comunicación con servidor FTP	8
Diseño, desarrollo y ensamblaje del PCB	36
Ampliación del ensamblaje de hardware: Se reajustó la disposición de las ruedas, se diseñó y agregó un soporte para las baterías, se diseñó y agregó un soporte para la fijación de los PCB diseñados.	5
Redacción de informes	19
Total horas invertidas	210



Centro de Técnicas Analógico-Digitales (CETAD)

Bibliografía

naylampmechatronics. MÓDULO ESP-32 con cámara OV2640 WIFI-SERIAL (2025). Disponible en internet:

<https://naylampmechatronics.com/espressif-esp/700-esp32-cam-con-camara-ov2640-esp32-wifi-base-ch340.html>

Consultado el 11/3/2025.

Hoja de datos:

<https://naylampmechatronics.com/img/cms/Datasheets/ESP32-CAM%20Product%20Specification.pdf>

Consultado el 11/3/2025

naylampmechatronics. Driver Puente H L298N 2A (2025). Disponible en internet:

<https://naylampmechatronics.com/drivers/11-driver-puente-h-l298n.html>

Consultado el 11/3/2025.

Hoja de datos:

<https://www.st.com/resource/en/datasheet/l298.pdf>

Consultado el 11/3/2025

naylampmechatronics. Motor DC TT 6V/200RPM (2025). Disponible en internet:

<https://naylampmechatronics.com/motores-dc/606-motor-dc-tt-6v200rpm.html>

Consultado el 11/3/2025.

Todomicro. Driver ULN2003A para motor Paso A Paso unipolar. Disponible en internet:

<https://www.todomicro.com.ar/controladores-de-motor/796-driver-uln2003-para-motor-paso-a-paso-unipolar.html>

Consultado el 11/3/2025.

naylampmechatronics. Motor pap (2025). Disponible en internet:



Centro de Técnicas Analógico-Digitales (CETAD)

<https://naylampmechatronics.com/motores-pap-steppers/365-motor-pap-28byj-48-5v.html>

Consultado el 11/3/2025.

Hoja de datos:

<https://robocraft.ru/files/datasheet/28BYJ-48.pdf>

Consultado el 11/3/2025

LAB1 Technologies. TF-LC02 LiDAR - Documentación y Ejemplos. Disponible en internet:

<https://soporte.lab1.tech/knowledgebase.php?article=12>

Consultado el 11/3/2025.

Especificaciones EDU-CIAA-NXP v1.1 Board - 2019-01-03 v5r0.pdf

Disponible en internet:

https://github.com/epernia/firmware_v3/blob/master/documentation/CIAA_Boards/NXP_LPC4337/EDU-CIAA-NXP/EDU-CIAA-NXP%20v1.1%20Board%20-%202019-01-03%20v5r0.pdf

Consultado el 11/3/2025.

Hobbytronica. Fuente LM2596 Step-down DC-DC 1,23-35v 3a Voltímetro Display. Disponible en internet:

https://www.hobbytronica.com.ar/MLA-901399057-fuente-lm2596-step-down-dc-dc-123-35v-3a-voltimetro-display-_JM

Consultado el 11/3/2025.



Centro de Técnicas Analógico-Digitales (CETAD)

Anexos

Esquemático.

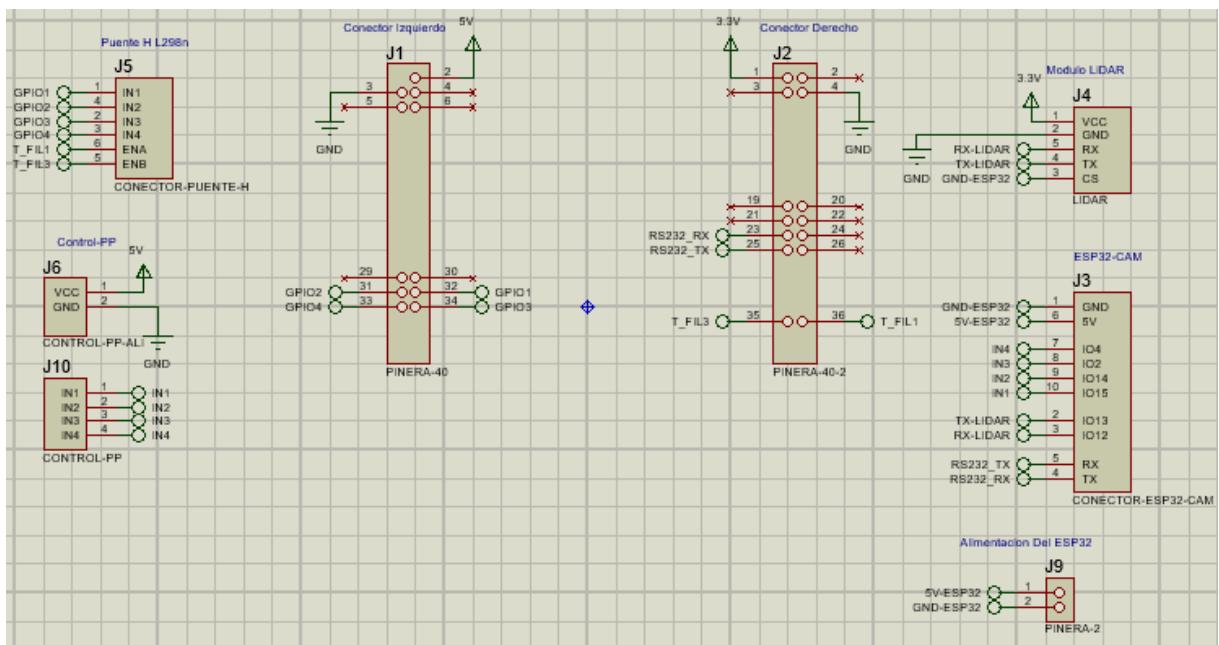


Figura A1. Esquemático completo de las conexiones a la EDU-CIAA.

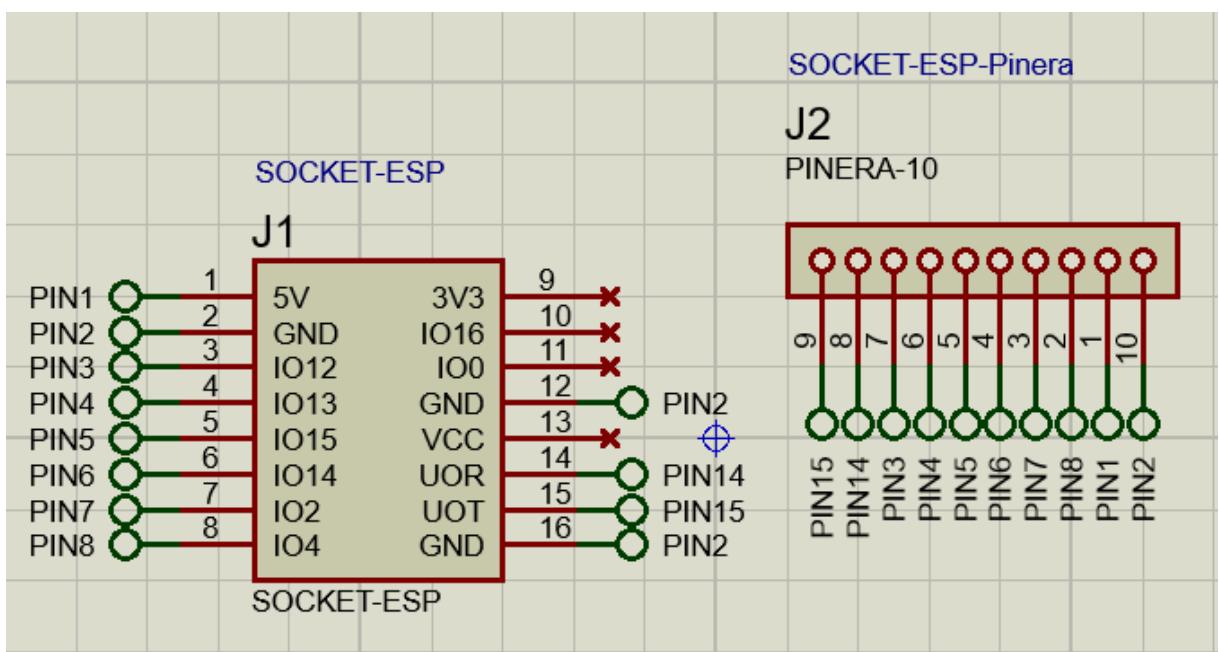


Figura A2. Esquemático completo del ESP32-CAM



Centro de Técnicas Analógico-Digitales (CETAD)

PCB

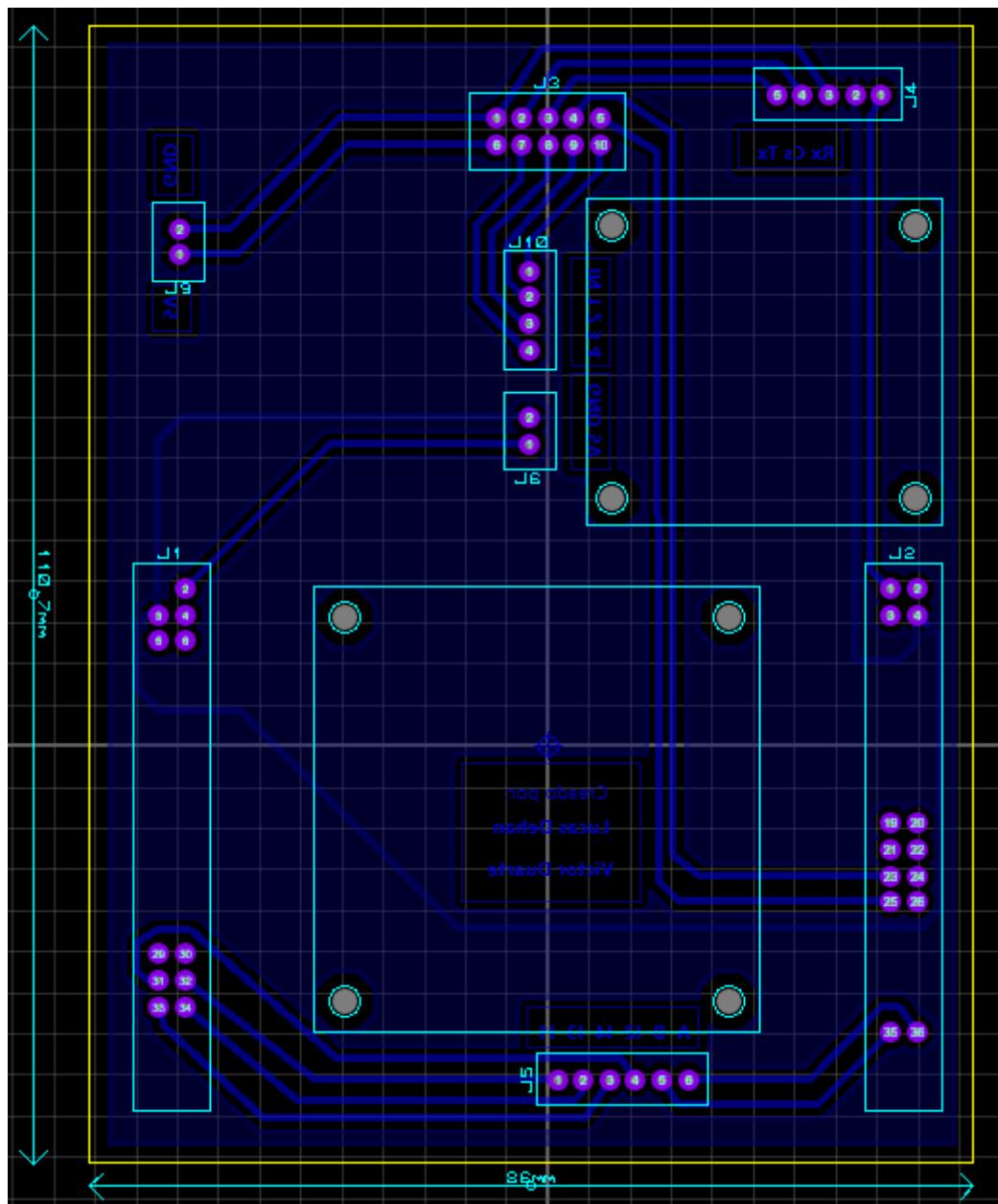


Figura A3. Diagrama del PCB completo de la EDU-CIAA en la vista “PCB Layout” con sus medidas aproximadas.



Centro de Técnicas Analógico-Digitales (CETAD)

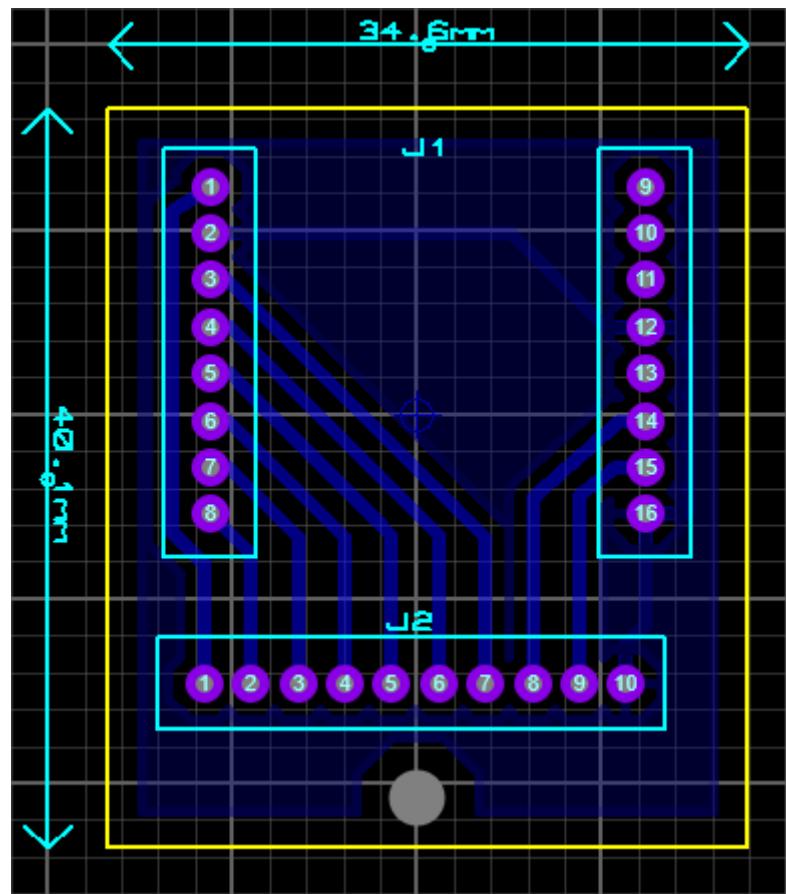


Figura A4. Diagrama del PCB completo del ESP32-CAM en la vista “PCB Layout” con sus medidas aproximadas.



Centro de Técnicas Analógico-Digitales (CeTAD)

Agradecimientos

Expresamos nuestro más sincero agradecimiento al equipo del CeTAD, en especial a los profesores **Walter Aróstegui, Leonardo Navarría y Jorge Rafael Osio**, por su dedicación, orientación y el valioso apoyo brindado durante el desarrollo de este trabajo, así como por sus invaluables aportes e ideas.

Asimismo, extendemos nuestro reconocimiento a nuestro compañero y amigo **Constantino Palacio**, cuya colaboración y disposición nos permitieron contar con las herramientas necesarias para la utilización del servidor FTP.

De igual manera, queremos agradecer al staff del ATEI en especial a **Alejandro Epifanio** por su apoyo fundamental en la creación del PCB, cuya contribución fue clave para el desarrollo de este proyecto.

A todos ellos, nuestro más profundo agradecimiento.