Grado en Ingeniería Electrónica Industrial y Automática

TRABAJO FIN DE GRADO

UNIVERSIDAD DE ALMERIA

ESCUELA SUPERIOR DE INGENIERÍA

Diseño, montaje y control de una maqueta de planta solar de receptor central inteligente

Curso 2024/2025

Alumno/a:

Luis Ignacio Jiménez Martínez

Director/es:

Manuel Berenguel Soria

Javier Bonilla Cruz

UNIVERSIDAD DE ALMERÍA

ESCUELA SUPERIOR DE INGENIERÍA

GRADO EN INGENIERÍA ELECTRÓNICA INDUSTRIAL Y AUTOMÁTICA

TRABAJO FIN DE GRADO

Diseño, montaje y control de una maqueta de planta solar de receptor central inteligente

---

Alumno:	Luis Ignacio Jiménez Martínez
Director:	Manuel Berenguel Soria
Codirector:	Javier Bonilla Cruz
Fecha:	5 de junio de 2025

Dedicado a mis padres y a mi hermana, por su amor, su apoyo incondicional y por ser mi mayor ejemplo de constancia y entrega.

A mis abuelos, que ya no están, pero cuya memoria y enseñanzas siguen presentes.

A toda mi familia, por su cariño, su paciencia y por estar siempre ahí.

Y a todas las personas que han recorrido este camino conmigo, compartiendo esfuerzo, aprendizajes y momentos que guardaré siempre.

Este trabajo también es vuestro.

Índice general

Agradecimientos

Al director **Manuel Berenguel Soria** y codirectores **Javier Bonilla Cruz** y **José Antonio Carballo López** por su dedicación y asesoramiento. Su experiencia y compromiso han sido fundamentales para alcanzar los objetivos.

Y a la **Plataforma Solar de Almería (PSA)** por su colaboración, facilitando recursos que han sido claves para su creación.

Resumen

En el presente trabajo se ha desarrollado un prototipo funcional que simula el comportamiento de una planta solar de concentración con tecnología de torre, tomando como referencia el modelo de funcionamiento de la Plataforma Solar de Almería (PSA). El proyecto ha sido concebido con un enfoque didáctico y técnico, combinando electrónica embebida, diseño mecánico, visión artificial y control automático, integrando todos los conocimientos adquiridos a lo largo del grado.

El prototipo se compone de dos bloques principales: la torre con un receptor y un heliostato. La torre, diseñada en SolidWorks e impresa en 3D, incorpora una placa solar fotovoltaica en su parte superior, cuya señal se mide y visualiza mediante un sistema basado en Arduino MEGA 2560 y una pantalla táctil TFT. Esta interfaz permite al usuario consultar en tiempo real la tensión generada y visualizar gráficamente su evolución, simulando la radiación captada en una planta real.

Por otro lado, se ha implementado un heliostato funcional controlado por una Raspberry Pi 4, encargado de reflejar la luz solar sobre el receptor. Para lograrlo, se ha empleado una cámara y un modelo de inteligencia artificial entrenado por la propia PSA, capaz de detectar el foco de luz (simulando el sol) y la ubicación del panel fotovoltaico (simulando el receptor en la torre). A partir de esa información, el sistema gobierna dos motores de corriente continua a través de una controladora L298N, ajustando en tiempo real la orientación del espejo.

El conjunto del sistema ha sido validado mediante pruebas funcionales en condiciones reales, verificando la correcta alineación del haz reflejado, la respuesta del sistema ante variaciones de luminosidad y la estabilidad del control durante períodos prolongados. Además, se han implementado controles manuales que permiten verificar el funcionamiento del heliostato y facilitar tareas de mantenimiento.

Este proyecto demuestra la viabilidad de replicar el funcionamiento básico de una planta termosolar a pequeña escala utilizando tecnologías accesibles. A su vez, representa una herramienta útil tanto para la divulgación científica y técnica como para el análisis experimental de conceptos clave en energía solar, inteligencia artificial, automatización industrial y visión artificial.

Palabras clave: *Heliostato, Planta solar de receptor central, modelo 3D, control automático, inteligencia artificial*

Abstract

In this project, a functional model has been developed to simulate the behavior of a central tower solar power plant, taking as reference the operational model of the Plataforma Solar de Almería (PSA). The project has been designed with both educational and technical purposes in mind, combining embedded electronics, mechanical design, computer vision, and automatic control—integrating all the knowledge acquired throughout the engineering degree.

The prototype consists of two main components: a tower with a receiver, and a heliostat. The tower, designed in SolidWorks and 3D printed, incorporates a photovoltaic solar panel at the top. Its output signal is monitored and displayed using an Arduino MEGA 2560-based system and a TFT touchscreen. This interface allows users to consult the generated voltage in real time and view its evolution graphically, simulating the radiation received in a real solar facility.

On the other hand, a functional heliostat has been implemented and controlled by a Raspberry Pi 4, responsible for reflecting sunlight toward the receiver. To achieve this, a camera and an artificial intelligence model—trained by the PSA itself—are used to detect both the sun's position and the location of the receiver on the tower. Based on this information, the system controls two DC motors via an L298N motor driver, dynamically adjusting the mirror's orientation in real time.

The complete system has been validated through functional tests under real conditions, verifying the correct alignment of the reflected beam, the system's response to changes in light intensity, and the stability of the control loop during extended operation. In addition, manual controls have been implemented to allow verification of the heliostat's operation and to facilitate maintenance tasks.

This project demonstrates the feasibility of replicating the basic operation of a solar thermal power plant at small scale using accessible technologies. Furthermore, it serves as a valuable tool for scientific and technical outreach as well as for the experimental analysis of key concepts in solar energy, artificial intelligence, industrial automation, and artificial vision.

Keywords: *Heliostat, Central receiver solar power plant, 3D model, Automatic control, Artificial intelligence*

Siglas y acrónimos

ANN	Red Neuronal Artificial (*Artificial Neural Network*)
ARM	Grupo de Investigación TEP-197, Automática, Robótica y Mecatrónica
CAD	Diseño Asistido por Ordenador (*Computer-Aided Design*)
COCO	Objetos comunes en contexto (*Common Objects in Context*)
CPU	Unidad central de procesamiento (*Central Processing Unit*)
CSI	Interfaz serie para cámaras (*Camera Serial Interface*)
DC	Corriente continua (*Direct Current*)
CSP	Energía solar concentrada (*Concentrated Solar Power*)
DSI	Interfaz serie de pantalla (*Display Serial Interface*)
FDM	Modelado por deposición fundida (*Fused Deposition Modeling*)
FIFO	Primero en entrar, Primero en salir (*First In, First Out*)
GPIO	Entrada/Salida de Propósito General (*General Purpose Input/Output*)
GPS	Sistema de Posicionamiento Global (*Global Positioning System*)
IA	Inteligencia Artificial (AI, *Artificial Inteligence*)
IDE	Entorno de desarrollo integrado (*Integrated Development Environment*)
LCD	Pantalla de cristal líquido (*Liquid Crystal Display*)
OS	Sistema operativo (*Operating system*)
PLA	Ácido poliláctico (*Polylactic Acid*)

PSA	Plataforma Solar de Almería
RAM	Memoria de acceso aleatorio (*Random Access Memory*)
RPM	Revoluciones por minuto
STL	Lenguaje de triángulo estándar (*Standard Triangle Language*)
TFG	Trabajo Fin de Grado
TFT	Transistor de película delgada (*Thin Film Transistor*)
TIC	Tecnologías de la Información y la Comunicación
TOPS	Teraoperaciones por Segundo (*Tera Operations Per Second*)
TPU	Unidad de procesamiento tensorial (*Tensor Processing Unit*)
UAL	Universidad de Almería
USB	Bus Serie Universal (*Universal Serial Bus*)

Índice de figuras

Índice de tablas

DISEÑO, MONTAJE Y CONTROL DE UNA MAQUETA DE PLANTA SOLAR DE RECEPTOR CENTRAL INTELIGENTE

Capítulo 1

Introducción

1.1. Introducción e interés

La creciente necesidad de fuentes energéticas sostenibles ha impulsado el desarrollo de tecnologías basadas en energías renovables, entre las cuales la energía solar térmica de concentración ocupa un papel destacado. Este tipo de tecnología se basa en la captación y concentración de la radiación solar para su posterior conversión en energía térmica, y ha sido ampliamente estudiada y desarrollada en instalaciones de referencia como la Plataforma Solar de Almería (PSA), uno de los centros de investigación más avanzados del mundo en este campo.

El presente proyecto tiene como objetivo el diseño, desarrollo e implementación de un prototipo funcional que se comporta del mismo modo que una torre solar de concentración, inspirada directamente en la arquitectura y principios operativos de la PSA. Para ello, se ha desarrollado una réplica a escala de la torre con receptor central, la cual ha sido modelada tridimensionalmente mediante el software CAD SolidWorks [1] y fabricada mediante tecnología de impresión 3D.

El sistema se ha dotado de una infraestructura de control distribuido que incluye un microcontrolador Arduino MEGA 2560 encargado de gestionar la interfaz de usuario a través de una pantalla táctil TFT, así como la monitorización de una placa solar ubicada en la cima de la torre. Esta placa actúa como receptor de la radiación concentrada.

El sistema de seguimiento solar ha sido implementado mediante un heliostato operado por una Raspberry Pi 4. Dicha unidad procesa la imagen capturada por una cámara conectada, analizando en tiempo real la posición del sol y la orientación del haz reflejado. A través de algoritmos de visión artificial y control por realimentación, el sistema gobierna dos motores conectados a una montura motorizada, ajustando dinámicamente la orientación del espejo del heliostato para garantizar que la luz solar se refleje directamente sobre la placa receptora de la

torre.

Este trabajo se enmarca dentro del contexto académico del Trabajo Fin de Grado (TFG) del Grado en Ingeniería Electrónica Industrial y Automática, integrando conocimientos y competencias en automatización, control electrónico, visión artificial, modelado y fabricación aditiva. El objetivo fundamental es ofrecer una representación didáctica y funcional de los principios que rigen las instalaciones termosolares de concentración, fomentando el aprendizaje aplicado y la innovación tecnológica a escala prototípica.

1.2. Motivación del TFG

El presente proyecto nace de la confluencia entre el interés personal por las tecnologías energéticas sostenibles y la necesidad académica de integrar los conocimientos adquiridos a lo largo del Grado en Ingeniería Electrónica Industrial y Automática. En particular, se ha buscado desarrollar un sistema funcional que permita representar, a escala reducida, los principios fundamentales de funcionamiento de una central termosolar de torre, además de una novedosa metodología de control basada en IA y visión artificial, tomando como referencia la PSA, centro pionero en investigación sobre energías renovables.

La elección de esta temática responde a varias motivaciones. Por un lado, la relevancia creciente de las energías renovables en el panorama energético global impulsa a los ingenieros a adquirir competencias técnicas que permitan comprender, implementar y mejorar este tipo de sistemas. Por otro lado, se ha pretendido desarrollar un proyecto multidisciplinar que combine electrónica, programación, diseño mecánico y fabricación aditiva, reflejando así el perfil integrador y versátil que se espera de un ingeniero del siglo XXI.

Además, se ha buscado una aplicación práctica y educativa: mediante la construcción de un prototipo funcional con elementos de bajo coste y tecnologías accesibles (Arduino, Raspberry Pi, impresión 3D), se logra un recurso didáctico que puede ser utilizado en contextos formativos para demostrar conceptos complejos como el seguimiento solar, la concentración de energía, la visión artificial aplicada o el control automático en sistemas distribuidos.

Finalmente, esta propuesta constituye un desafío técnico estimulante y una oportunidad para explorar tecnologías actuales como el reconocimiento visual con inteligencia artificial (IA), la integración hardware-software y el diseño de sistemas mecatrónicos, elementos clave en el ámbito de la automatización industrial. Todo ello ha servido como base motivacional sólida para desarrollar un proyecto con valor académico, técnico y social.

1.3. Objetivos

El presente TFG tiene como propósito fundamental el diseño y desarrollo de un prototipo a escala funcional que represente el principio de funcionamiento de una planta solar de concentración (CSP) con receptor central. El prototipo integra una torre de captación y un sistema de heliostato automatizado, con el fin de simular el proceso de concentración de radiación solar sobre un receptor fijo.

Los objetivos específicos que estructuran este proyecto son los siguientes:

- **Diseño y construcción del prototipo:** desarrollar un prototipo funcional de una instalación termosolar de torre, que represente fielmente los elementos esenciales de una planta CSP de receptor central, incluyendo una torre receptora y un heliostato orientable.
- **Integración de electrónica en la torre:** implementar un sistema de adquisición y visualización de datos utilizando microcontroladores de bajo coste (Arduino), en combinación con una pantalla TFT LCD. Este sistema permitirá:
  - Medir la señal de salida generada por una placa solar fotovoltaica colocada en la parte superior de la torre, representando la captación de energía solar concentrada.
  - Mostrar en tiempo real variables de interés como el nivel de tensión o la intensidad lumínica percibida en el receptor.
- **Desarrollo de un sistema de control autónomo para el heliostato:** diseñar e implementar un sistema inteligente de seguimiento solar utilizando una Raspberry Pi y técnicas de visión artificial. Este subsistema se encargará de:
  - Captar imágenes mediante una cámara integrada para identificar la posición del sol y la ubicación del receptor solar en la torre.
  - Controlar en tiempo real la orientación del heliostato, mediante el accionamiento de dos motores, de forma que el reflejo del haz solar incida continuamente sobre la placa receptora.
- **Verificación del sistema:** realizar ensayos experimentales que permitan validar el correcto funcionamiento del sistema, evaluando la eficacia del control del heliostato y la consistencia de la captación de radiación en la torre.

Estos objetivos persiguen no solo una demostración técnica de los conceptos involucrados, sino también la creación de una herramienta didáctica que pueda emplearse para la divulgación y enseñanza de principios asociados a las energías renovables, la automatización industrial y la inteligencia artificial embebida.

1.4. Contexto y bibliografía principal

1.4.1. Tecnología de concentración solar (CSP)

La energía solar de concentración (CSP, por sus siglas en inglés: *Concentrated Solar Power*) es una tecnología que permite aprovechar la radiación solar directa concentrándola mediante sistemas ópticos sobre un receptor térmico, donde se genera calor que puede utilizarse para la producción de electricidad. A diferencia de los sistemas fotovoltaicos, que convierten la radiación directamente en energía eléctrica, las plantas CSP producen calor de proceso, que puede almacenarse o convertirse posteriormente en energía mediante ciclos termodinámicos.

Existen varias configuraciones de plantas de concentración solar, entre las que se destacan las de torre central, captadores cilindro-parabólicos, sistemas de disco parabólico y reflectores de Fresnel lineales. Entre ellas, las plantas de torre con receptor central se consideran una de las más prometedoras por su capacidad para alcanzar temperaturas superiores a 550 °C y su eficiencia térmica.

El principio básico de una planta de torre consiste en utilizar un campo de heliostatos, es decir, reflectores solares que conforman superficies esféricas con seguimiento a dos ejes, que siguen el movimiento aparente del sol y concentran su radiación sobre un receptor ubicado en la parte superior de una torre. Este receptor capta la energía solar y transfiere el calor a un fluido térmico, que posteriormente se utiliza en un generador de vapor y una turbina.

Uno de los referentes internacionales en investigación de tecnología CSP es la PSA, situada en el desierto de Tabernas. En sus instalaciones se han desarrollado numerosos prototipos y sistemas de prueba, incluyendo heliostatos de nueva generación, receptores avanzados y algoritmos de control. A escala internacional, la tecnología CSP sigue evolucionando en aspectos como eficiencia óptica, reducción de costes y automatización de sistemas de seguimiento [2].

Este proyecto toma como referencia conceptual esta configuración de torre central para desarrollar un sistema a escala reducida que reproduzca un seguimiento solar, con el fin de demostrar su funcionamiento mediante visión artificial y control automático.

1.4.2. Sistemas de seguimiento solar y heliostatos

Los heliostatos son componentes clave en las plantas solares de torre central. Están formados por espejos con superficies esféricas capaces de orientarse sobre dos ejes para seguir la trayectoria del sol y reflejar su radiación hacia un receptor fijo. Para lograr este seguimiento, se emplean sistemas de control que calculan la posición del sol y ajustan constantemente la orientación del espejo.

En el ámbito de la investigación, se han desarrollado diferentes técnicas de seguimiento,

desde modelos matemáticos basados en coordenadas solares hasta métodos de visión artificial y aprendizaje automático [3, 4]. El uso de heliostatos inteligentes permite mejorar la precisión del enfoque, reducir errores angulares y optimizar la captación de energía.

En [4], se propone un sistema de seguimiento solar de bajo coste desarrollado con hardware abierto, orientado a aplicaciones educativas. Otros trabajos, como el de [5], exploran la aplicación de redes neuronales profundas y visión por computador para realizar el seguimiento del sol sin necesidad de sensores adicionales. La implementación de estos enfoques ha permitido avanzar hacia heliostatos autónomos capaces de aprender y adaptarse a condiciones variables de iluminación y meteorología.

1.4.3. Visión artificial y aprendizaje automático aplicados al control solar

La visión artificial ha emergido como una herramienta potente para detectar, localizar y seguir objetos de interes. Al integrar cámaras digitales y algoritmos de procesamiento de imagen, es posible identificar con precisión la ubicación del sol o de un punto objetivo (como el receptor solar en una torre) en tiempo real.

El trabajo presentado en [5] demuestra que la visión por computador combinada con aprendizaje profundo puede reemplazar los sistemas tradicionales basados en sensores, ofreciendo mayor flexibilidad y adaptabilidad. Además, [3] expone cómo se puede aplicar el aprendizaje por refuerzo para optimizar el posicionamiento de heliostatos, mejorando su rendimiento energético frente a métodos clásicos.

En este proyecto, se ha utilizado una Raspberry Pi equipada con cámara y un modelo de inferencia entrenado por la PSA, capaz de identificar tanto una fuente de luz (imitando al sol) como una placa fotovoltaica (imitando el receptor) en la torre a partir del análisis de imágenes. Esta información se emplea para ajustar automáticamente los motores del heliostato, reflejando la luz hacia el receptor.

1.4.4. Prototipos experimentales y plataformas educativas

El desarrollo de prototipos a pequeña escala permite validar algoritmos, probar componentes y formación. En [4], se describe un prototipo funcional de seguimiento solar desarrollado con Arduino y sensores de luz, pensada para uso en entornos formativos. Asimismo, en [6] se presenta un banco de pruebas completo que simula una planta solar de torre, diseñado para evaluar técnicas de control avanzadas.

Este tipo de iniciativas facilita la transferencia del conocimiento y permite escalar soluciones hacia aplicaciones reales. El presente trabajo se alinea con esta filosofía, proporcionando un entorno experimental en el que convergen visión artificial, electrónica de potencia, control e

impresión 3D.

1.5. Resumen de resultados

Tras la fase de diseño, implementación y puesta en marcha del prototipo, se han obtenido una serie de resultados que permiten validar funcional y técnicamente los objetivos planteados.

En primer lugar, se ha logrado construir una torre solar a escala equipada con una placa solar fotovoltaica en su parte superior, capaz de detectar de forma precisa la radiación solar incidente. La estructura de la torre ha sido diseñada íntegramente en SolidWorks y fabricada mediante impresión 3D con un alto grado de fidelidad respecto al modelo conceptual de la PSA. La integración de componentes electrónicos en la torre, tales como un microcontrolador Arduino MEGA 2560 y una pantalla TFT táctil, ha permitido visualizar en tiempo real la magnitud de la tensión generada por la placa solar, representando de forma tangible el efecto de concentración solar.

En paralelo, se ha desarrollado un sistema de control autónomo para el heliostato basado en una Raspberry Pi 4. Este sistema, utilizando visión artificial y una red neuronal [7] con una cámara conectada, ha demostrado ser capaz de detectar la posición del sol y la ubicación del receptor en la torre. Mediante el accionamiento de dos motores, el sistema ajusta de forma continua la orientación del concentrador para reflejar la luz solar sobre la placa receptora, logrando un seguimiento dinámico eficaz.

Los ensayos realizados en condiciones de iluminación reales han mostrado una respuesta estable y coherente del sistema: el heliostato consigue seguir el movimiento del foco de luz (imitando al movimiento aparente del sol) con precisión suficiente para mantener la proyección del haz luminoso en la zona activa del receptor durante intervalos prolongados. La validación del comportamiento del sistema se ha llevado a cabo mediante la monitorización continua de la tensión en la placa solar y la observación directa de la trayectoria del haz reflejado.

En resumen, el prototipo desarrollado cumple con los objetivos propuestos, reproduce de forma coherente los principios físicos y operativos de una planta termosolar de torre, y constituye una plataforma funcional válida tanto para demostraciones didácticas como para el estudio de técnicas de automatización y control embebido en entornos reales.

1.6. Fases de desarrollo y planificación

A continuación se detallan las distintas fases seguidas durante el desarrollo del proyecto, las cuales se estructuraron de forma secuencial para abordar de manera integral los aspectos mecánicos, electrónicos y software del sistema automatizado de seguimiento solar para heliostatos.

1. **Revisión bibliográfica (A) [25.2h].**

   Esta fase inicial tuvo como objetivo sentar las bases teóricas del proyecto, mediante la recopilación y análisis de literatura técnica relacionada con tecnologías de seguimiento solar, control de motores, visión artificial y procesamiento.

2. **Diseño e impresión de la torre (B) [31.5h].** Se diseñó la estructura de soporte del sistema utilizando herramientas CAD como SolidWorks, teniendo en cuenta criterios de estabilidad, precisión mecánica y facilidad de ensamblaje.

3. **Programación Arduino (C) [52h].**

   Se desarrolló el firmware de la placa Funduino Mega 2560 para gestionar la pantalla TFT táctil de 2,8" y la lectura de la placa fotovoltaica. La interfaz permite visualizar menús, datos de tensión y potencia, y acceder a una gráfica en tiempo real.

4. **Montaje físico de la torre (D) [21h].** En esta fase se procedió al ensamblaje físico de los componentes estructurales y electrónicos en la estructura impresa. Se instaló y posicionó la placa solar y la pantalla táctil de manera segura, verificando la correcta alineación y realizando el tendido preliminar del cableado necesario para la alimentación y control del sistema.

5. **Impresión del heliostato (E) [15.2h].**

   En esta fase, partiendo del diseño original ya realizado e impreso del heliostato por la PSA, se rediseñaron e imprimieron únicamente aquellas piezas necesarias para corregir errores dimensionales o interferencias detectadas durante el montaje inicial. Las modificaciones se centraron principalmente en soportes, acoplamientos y elementos auxiliares que no encajaban correctamente o dificultaban el ensamblaje final.

6. **Programación de la Raspberry Pi (F) [106.8h].** Esta fase fue la más intensiva en tiempo, dado que abarca toda la lógica de control, visión e interfaz de usuario. Se impementó un sistema de captura de vídeo en tiempo real, ejecución de una red neuronal de detección de objetos, visualización de streaming de cámara en una pantalla, y actuación sobre los motores del heliostato.

7. **Montaje final del heliostato (G) [7.6h].** Una vez desarrolladas todas las partes funcionales, se integró el sistema completo: partes impresas en 3D, motores, controladores, Raspberry Pi, cámara y pantalla táctil.

   Se consolidó el cableado definitivo con una fuente de alimentación externa de 12 V para los motores, y 5 V para la lógica de control. Se verificó el comportamiento conjunto del sistema y se preparó para las pruebas finales.

8. **Pruebas y validación (H) [22h].** Se llevaron a cabo ensayos de funcionamiento en condiciones reales para validar el comportamiento del sistema. Se analizaron:

   - Tiempo de respuesta del sistema entre detección y corrección.

- Estabilidad en el seguimiento continuo del sol.
- Precisión del centrado respecto al objetivo.
- Robustez frente a interferencias y continuidad operativa.

9. **Documentación final (I) [43.7h].** La fase final consistió en la redacción de la memoria del TFG. El documento fue estructurado siguiendo las pautas de la Escuela de Ingeniería, incorporando referencias bibliográficas en estilo IEEE, diagramas de Gantt y fotografías del montaje final, todo con el objetivo de respaldar de forma clara y profesional los resultados obtenidos.

Tabla 1.1. Planificación temporal

		ACTIVIDAD								
MES	SEMANA	A	B	C	D	E	F	G	H	I
FEBRERO	3									
	4									
MARZO	1									
	2									
	3									
	4									
	5									
ABRIL	1									
	2									
	3									
	4									
MAYO	1									
	2									
	3									
	4									
JUNIO	1									

1.7. Competencias empleadas en este trabajo

1.7.1. Competencias básicas, generales y transversales

- **CB1:** Demostrar poseer y comprender conocimientos en un área de estudio que parte de la base de la educación secundaria general, y se suele encontrar a un nivel que, si bien se apoya en libros de texto avanzados, incluye también algunos aspectos que implican conocimientos procedentes de la vanguardia de su campo de estudio.
  Aplicación en el TFG: Aplicada durante la revisión bibliográfica, donde se integraron conceptos avanzados de seguimiento solar, visión artificial y control embebido.

- **CB2:** Saber aplicar los conocimientos a su trabajo o vocación de una forma profesional y poseer las competencias que suelen demostrarse por medio de la elaboración y defensa de argumentos y la resolución de problemas dentro de su área de estudio.
  Aplicación en el TFG: Demostrada en la programación y montaje del sistema, aplicando conocimientos teóricos a la práctica con resultados funcionales.

- **CB3:** Tener la capacidad de reunir e interpretar datos relevantes (normalmente dentro de su área de estudio) para emitir juicios que incluyan una reflexión sobre temas relevantes de índole social, científica o ética.
  Aplicación en el TFG: Al interpretar los datos medidos por sensores y valorar la efectividad del sistema en condiciones reales.

- **CB4:** Poder transmitir información, ideas, problemas y soluciones a un público tanto especializado como no especializado.
  Aplicación en el TFG: Durante la elaboración de la memoria y la presentación de resultados técnicos a tutores.

- **CB5:** Haber desarrollado aquellas habilidades de aprendizaje necesarias para emprender estudios posteriores con un alto grado de autonomía.
  Aplicación en el TFG: Al adquirir de forma autónoma nuevas herramientas como libcamera, Coral TPU o interfaces táctiles.

1.7.2. Competencias transversales de la UAL

- **UAL2:** Utilizar las Técnicas de Información y Comunicación (TIC) como una herramienta para la expresión y la comunicación, para el acceso a fuentes de información, como medio de archivo de datos y documentos, para tareas de presentación, para el aprendizaje, la investigación y el trabajo cooperativo.
  Aplicación en el TFG: Al desarrollar interfaces en Python, usar herramientas CAD y controlar sistemas mediante software.

- **UAL3:** Capacidad para identificar, analizar, y definir los elementos significativos que constituyen un problema para resolverlo con rigor.
  Aplicación en el TFG: Durante la depuración de errores, adaptación de piezas y solución de interferencias mecánicas.

- **UAL4:** Comprender y expresar con claridad y oportunidad las ideas, conocimientos, problemas y soluciones a un público más amplio, especializado o no especializado.
  Aplicación en el TFG: Al redactar documentación técnica clara y precisa.

- **UAL5:** Es el comportamiento mental que cuestiona las cosas y se interesa por los fundamentos en los que se asientan las ideas, acciones y juicios, tanto propios como ajenos.
  Aplicación en el TFG: Reflejada en la toma de decisiones técnicas durante el diseño y pruebas.

- **UAL6:** Integrarse y colaborar de forma activa en la consecución de objetivos comunes con otras personas, áreas y organizaciones, en contextos tanto nacionales como internacionales.
  Aplicación en el TFG: Mediante la coordinación con tutor/es y colaboración puntual en impresión 3D o adquisición de componentes.

- **UAL9:** Capacidad para diseñar, gestionar y ejecutar una tarea de forma personal.
  Aplicación en el TFG: Desarrollada al gestionar de forma autónoma todas las fases del proyecto.

1.7.3. Competencias específicas

- **E-CT1:** Capacidad para la redacción, firma y desarrollo de proyectos en el ámbito de la ingeniería industrial que tengan por objeto la construcción, reforma, reparación, conservación, demolición, fabricación, instalación, montaje o explotación de: estructuras, equipos mecánicos, instalaciones energéticas, instalaciones eléctricas y electrónicas, instalaciones y plantas industriales y procesos de fabricación y automatización.
  Aplicación en el TFG: Al realizar un proyecto completo de automatización de un heliostato con todos sus subsistemas.

- **E-CT3:** Conocimiento en materias básicas y tecnológicas, que les capacite para el aprendizaje de nuevos métodos y teorías, y les dote de versatilidad para adaptarse a nuevas situaciones.
  Aplicación en el TFG: Durante la incorporación de nuevas tecnologías no vistas directamente en clase (Coral, Python, Arduino).

- **E-CT4:** Capacidad de resolver problemas con iniciativa, toma de decisiones, creatividad, razonamiento crítico y de comunicar y transmitir conocimientos, habilidades y destrezas en el campo de la Ingeniería Industrial.
  Aplicación en el TFG: En la resolución de problemas prácticos de control, diseño y visión artificial.

- **E-CT5:** Conocimientos para la realización de mediciones, cálculos, valoraciones, tasaciones, peritaciones, estudios, informes, planes de labores y otros trabajos análogos.
  Aplicación en el TFG: Aplicada en la toma de medidas, cálculos de posicionamiento, y análisis de tiempos de respuesta.

- **E-CT6:** Capacidad para el manejo de especificaciones, reglamentos y normas de obligado cumplimiento.
  Aplicación en el TFG: Al consultar especificaciones de motores, pantallas, sensores y normas de conexionado.

- **E-CT7:** Capacidad de analizar y valorar el impacto social y medioambiental de las soluciones técnicas.

Aplicación en el TFG: En la elección de soluciones de bajo consumo y mínima intervención mecánica.

- **E-CT8:** Capacidad para aplicar los principios y métodos de la calidad.
  Aplicación en el TFG: Al aplicar mejoras incrementales en el sistema para optimizar su fiabilidad.
- **E-CT9:** Capacidad de organización y planificación en el ámbito de la empresa, y otras instituciones y organizaciones.
  Aplicación en el TFG: En la planificación temporal del proyecto y la gestión de tareas en paralelo.
- **E-CT10:** Capacidad de trabajar en un entorno multilingüe y multidisciplinar.
  Aplicación en el TFG: Al manejar documentación técnica en inglés y entornos de trabajo con herramientas de distintos dominios.
- **E-CT11:** Conocimiento, comprensión y capacidad para aplicar la legislación necesaria en el ejercicio de la profesión de Ingeniero Técnico Industrial.
  Aplicación en el TFG: Al considerar los requisitos eléctricos de seguridad en la alimentación y control.
- **E-CB3:** Conocimientos básicos sobre el uso y programación de los ordenadores, sistemas operativos, bases de datos y programas informáticos con aplicación en ingeniería.
  Aplicación en el TFG: En el uso de Raspberry Pi, Arduino, programación en Python y C++, y gestión de librerías.
- **E-CB5:** Capacidad de visión espacial y conocimiento de las técnicas de representación gráfica, tanto por métodos tradicionales de geometría métrica y geometría descriptiva, como mediante las aplicaciones de diseño asistido por ordenador.
  Aplicación en el TFG: Durante la interpretación de planos y diseño CAD de piezas en 3D.
- **E-CTEE5:** Conocimiento aplicado de instrumentación electrónica.
  Aplicación en el TFG: En la lectura y procesamiento de señales de sensores solares.
- **E-CTEE7:** Conocimiento y capacidad para el modelado y simulación de sistemas.
  Aplicación en el TFG: En la simulación del comportamiento de controladores y respuesta del sistema.
- **E-CTEE8:** Conocimientos de regulación automática y técnicas de control y su aplicación a la automatización industrial.
  Aplicación en el TFG: Al desarrollar un sistema de control de posición automática basado en visión artificial.
- **E-CTEE10:** Conocimiento aplicado de informática industrial y comunicaciones.
  Aplicación en el TFG: En la integración de la interfaz Python con hardware a través de GPIO y comunicación serie.

- **E-CTEE11:** Capacidad para diseñar sistemas de control y automatización industrial. *Aplicación en el TFG:* Diseñando la lógica de automatización completa para el seguimiento solar en dos ejes.

1.8. Estructura del TFG

El trabajo desarrollado se expone atendiendo a una estructura de 5 capítulos. A continuación, se exponen de forma resumida, en orden de presentación:

- Capítulo 1. Introducción: En este primer capítulo se ha contextualizado el proyecto, exponiendo la motivación que lo justifica, el entorno tecnológico y académico en el que se enmarca, así como los objetivos principales a alcanzar. Además, se incluye la planificación temporal seguida durante su desarrollo y un resumen de los resultados obtenidos, que permiten valorar el grado de cumplimiento de los objetivos propuestos.
- Capítulo 2. Materiales y métodos: En este capítulo se detallan los componentes electrónicos, software y herramientas utilizados en el desarrollo de la torre solar y el heliostato, incluyendo Arduino, Raspberry Pi, impresión 3D y visión artificial.
- Capítulo 3. Desarrollo de la Torre Solar: En este capítulo se detalla el desarrollo completo de la torre solar a escala, diseñada e impresa en 3D como réplica conceptual de la PSA. Se describe la integración electrónica basada en Arduino Mega, incluyendo la visualización de datos en pantalla táctil y los distintos modos operativos: numérico y gráfico. También se explica la adquisición de datos desde la placa solar y la lógica de interfaz del usuario. Finalmente, se documenta el montaje estructural y la programación embebida que permite su funcionamiento autónomo.
- Capítulo 4. Desarrollo del heliostato Automatizado: Este capítulo aborda el diseño e implementación del heliostato inteligente, encargado de reflejar la radiación solar hacia la torre. Se describen su arquitectura mecánica en 3D, el sistema de doble eje accionado por motores DC controlados desde una Raspberry Pi y una controladora L298N, y el modelo de visión artificial entrenado por la PSA. Se detalla el algoritmo de detección y centrado automático basado en IA, así como la interfaz manual con botones táctiles para control directo. También se explican las pruebas de validación realizadas y el comportamiento del sistema en condiciones reales.
- Capítulo 5. Conclusiones y trabajos futuros: El presente trabajo ha permitido diseñar y construir una maqueta funcional de una planta solar de torre a pequeña escala, integrando sistemas de IA, visión artificial y control automático de precisión. Se ha validado la capacidad del sistema para detectar y centrar la radiación solar sobre un receptor, reproduciendo el funcionamiento básico de un heliostato real. Como líneas futuras, se propone

incorporar técnicas avanzadas de seguimiento de objetos y probar modelos de detección más eficientes, así como mejorar la robustez del sistema ante condiciones cambiantes de iluminación.

Capítulo 2

Materiales y Métodos

Este capítulo recoge de forma sistemática todos los componentes, herramientas, tecnologías y entornos de desarrollo empleados en la realización del proyecto, diferenciando claramente entre los subsistemas de la maqueta: la torre solar y el heliostato inteligente.

2.1. Torre solar

La torre solar ha sido concebida como una réplica funcional a escala de la infraestructura central de una planta CSP de receptor central, simulando el comportamiento del receptor térmico mediante una placa solar fotovoltaica. Su construcción ha requerido tanto medios de fabricación digital como integración electrónica. A continuación se detallan los principales elementos utilizados:

- **Microcontrolador:** Arduino Mega 2560 (Funduino). Permite gestionar las lecturas de la placa solar, calcular la potencia generada y actualizar los modos de visualización en la pantalla TFT en tiempo real.

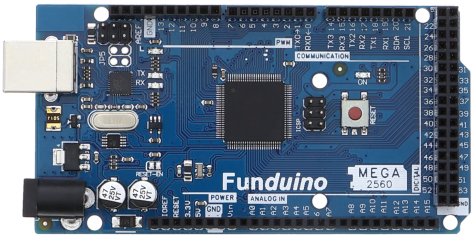

Figura 2.1. Arduino Mega 2560

- **Pantalla táctil:** TFT LCD de 2.8 pulgadas (Shield con chip ILI9341), utilizada para mostrar tanto datos numéricos (voltaje y potencia) como una gráfica en tiempo real. También permite interacción táctil con menús.

Figura 2.2. Pantalla TFT LCD

- **Placa solar:** Módulo fotovoltaico de 1 W utilizado para simular el receptor de torre, generando datos reales de radiación incidente.

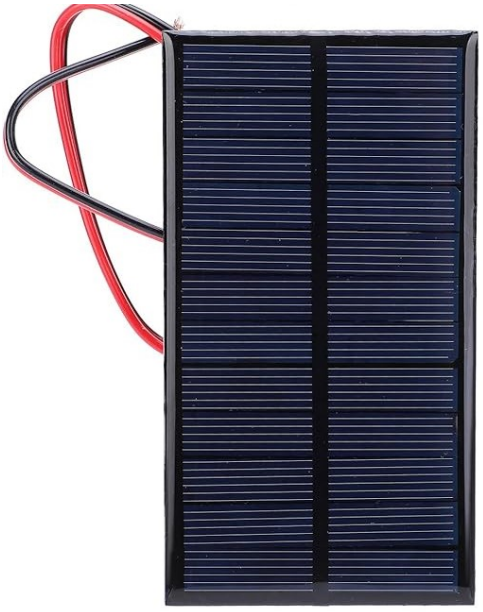

Figura 2.3. Panel fotovoltaico

- **Estructura:** Torre diseñada íntegramente en CAD (SolidWorks), impresa mediante tecnología FDM en impresoras 3D personales (Creality K1 y Ender 3 S1) utilizando filamento PLA estándar de 1.75 mm.

- **Entorno de desarrollo:** Arduino IDE versión 1.8.x, con compilación y carga directa sobre el Mega 2560.
- **Librerías utilizadas:**
  - `Adafruit_GFX.h` [8]
  - `MCUFRIEND_kbv.h` [9]
  - `TouchScreen.h` [10]

2.2. Heliostato inteligente

El subsistema del heliostato ha sido desarrollado con el objetivo de orientar un espejo curvo de forma automática, siguiendo el foco de luz (simulando el sol) y redirigiendo el haz solar hacia la torre, como ocurre en sistemas comerciales de torre central. Para ello, se ha empleado una arquitectura de computación embebida con capacidad de visión artificial y control en tiempo real.

- **Computadora:** Raspberry Pi 4 Modelo B, 4 GB RAM. Utilizada para ejecutar el modelo de detección de objetos y gestionar el control de motores; además de capturar streaming de la cámara, mostrarlo y detección táctil de la pantalla.

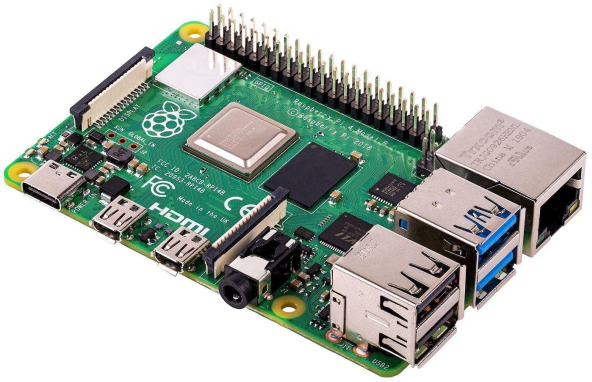

Figura 2.4. Raspberry Pi 4

- **Pantalla:** Pantalla táctil oficial de Raspberry Pi de 7 pulgadas conectada por DSI, utilizada para la interfaz de usuario con botones de control manual.

Figura 2.5. Raspberry Pi Screen

- **Cámara:** Raspberry Pi Camera Module v2.1 (Sony IMX219), empleada como sensor de visión para detectar tanto el foco de luz como la placa fotovoltaica. El sensor tiene una resolución nativa de 8 megapíxeles e incorpora un objetivo de enfoque fijo. En cuanto a imágenes fijas, la cámara es capaz de capturar imágenes estáticas de 3280 x 2464 píxeles y también admite vídeo de 1080p30, 720p60 y 640x480p90.

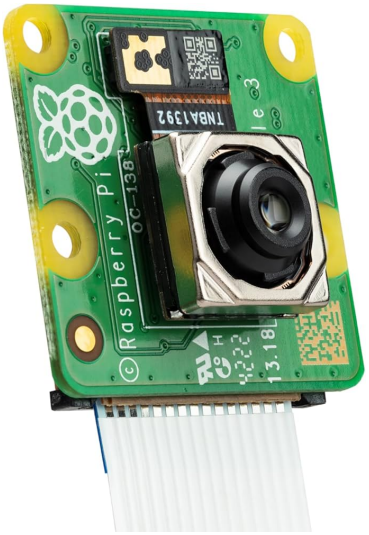

Figura 2.6. Raspberry Pi Camera

- **Controladora de motores:** Módulo doble puente *H* L298N, alimentado por fuente de 12 V DC y encargado de accionar dos motores en configuración doble eje [11].

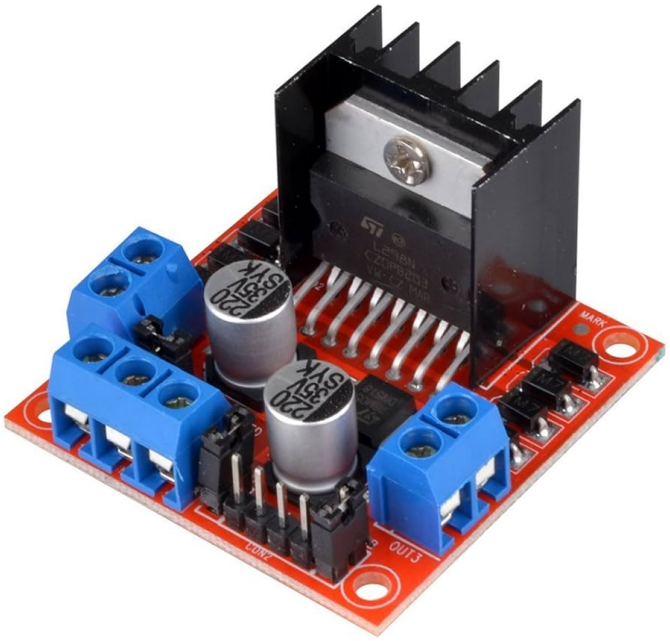

Figura 2.7. Controladora L298N

- **Motores:** Dos motores DC de 12 V, 2 RPM, con reductora metálica, utilizados para los ejes de orientación azimutal y elevación del heliostato.

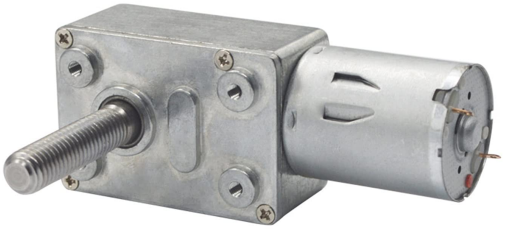

Figura 2.8. Motorreductor 12V

- **Fuente de alimentación:** Transformador con salida 12 V y capacidad suficiente para alimentar tanto controladora como motores.

Figura 2.9. Fuente de alimentación 12V

- **Visión artificial:** Acelerador Coral USB EdgeTPU, empleado para ejecutar el modelo entrenado por la PSA con baja latencia. Es capaz de realizar 4 billones de operaciones (teraoperaciones por segundo) (TOPS), utilizando 0,5 vatios por cada TOPS (2 TOPS por vatio)

Figura 2.10. USB Accelerator Coral

- **Tornillería y montaje:** Estructura diseñada en CAD y ensamblada con tornillos y tuercas M12 para garantizar robustez mecánica.

- **Librerías de Python utilizadas:**
  - `cv2 (OpenCV)` [12]
  - `numpy` [13]
  - `RPi.GPIO` [14]
  - `tkinter` [15]
  - `tflite_runtime.interpreter` [16]
- **Entorno de desarrollo:** Sistema operativo Raspberry Pi OS Lite, edición del código mediante VS Code y pruebas sobre entorno real con el Coral conectado por USB 3.0.

2.3. Recursos comunes y medios auxiliares

Ambas partes del proyecto comparten el uso de los siguientes elementos:

- **Impresoras 3D:** Creality K1 y Creality Ender 3 S1.
- **Filamento PLA:** Bobinas de 1.75 mm de diferentes colores utilizadas para la fabricación estructural.
- **Ordenador portátil HP Pavilion:** usado para modelado, desarrollo de firmware, simulación y edición de la memoria técnica.
- **Software de diseño CAD SolidWorks:** usado para el diseño mecánico de los componentes del sistema.

Capítulo 3

Desarrollo de la Torre Solar

Este capítulo describe en detalle el proceso de diseño, implementación y montaje de la torre solar a escala, la cual constituye uno de los elementos principales del sistema termosolar desarrollado. La torre integra tanto la estructura física como el sistema de adquisición de datos y visualización en tiempo real mediante una pantalla táctil, controlada por un microcontrolador Arduino MEGA 2560.

3.1. Diseño CAD e impresión 3D

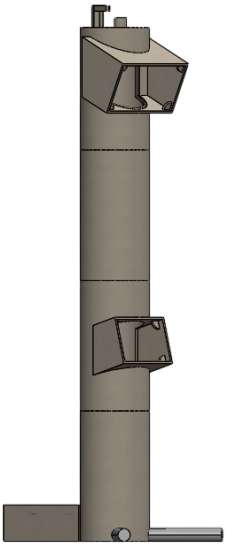

Figura 3.1. Modelo 3D de la torre solar diseñado en SolidWorks

La estructura de la torre ha sido diseñada utilizando el software SolidWorks, con el objetivo de emular la configuración de una torre de receptor central como las utilizadas en plantas solares de concentración. Se ha prestado especial atención a la integración de los componentes electrónicos, diseñando alojamientos específicos para la placa solar, la placa Arduino, la pantalla TFT y el cableado.

El diseño fue exportado en formato STL y fabricado mediante impresión 3D con tecnología FDM utilizando filamento PLA de 1.75 mm. Las piezas fueron ensambladas posteriormente utilizando tornillería estándar.

3.2. Montaje de componentes electrónicos

En el interior de la torre se ha instalado una placa solar fotovoltaica, encargada de simular el receptor de radiación solar concentrada. Esta placa se conecta a una entrada analógica del microcontrolador Arduino MEGA 2560, que interpreta la señal de tensión y la convierte en una medida proporcional de intensidad de radiación.

El sistema de visualización está compuesto por una pantalla TFT LCD de 2.8 pulgadas con funcionalidad táctil, basada en el controlador ILI9341. La pantalla se comunica con la placa Arduino a través de un bus paralelo de 8 bits y pines de control digitales, conectados en su totalidad a la placa mediante un shield compatible con la Mega 2560. [17]

Figura 3.2. Montaje electrónico de la torre con Arduino MEGA 2560

3.3. Librerías y configuración del entorno para la torre

El sistema electrónico de la torre solar fue desarrollado utilizando una placa Arduino MEGA 2560 programada en lenguaje C++ mediante el entorno Arduino IDE. Para controlar la pantalla táctil y gestionar la visualización de datos, fue necesaria la instalación y uso de varias librerías ampliamente utilizadas en proyectos con pantallas gráficas.

Librerías utilizadas

Las librerías principales empleadas fueron:

- **MCUFRIEND_kbv:** permite la comunicación con pantallas TFT tipo shield de 2.8 pulgadas basadas en el controlador ILI9341, utilizando un bus de datos paralelo.
- **Adafruit_GFX:** librería auxiliar que proporciona funciones para la visualización de elementos gráficos, como textos, líneas y botones, necesaria para representar correctamente los datos medidos.
- **TouchScreen.h:** utilizada para leer coordenadas táctiles en pantallas resistivas, permitiendo gestionar la interacción del usuario con los botones dibujados en pantalla.

Estas librerías fueron integradas en el proyecto a través del gestor de librerías del entorno Arduino IDE y funcionan de forma conjunta para permitir tanto la entrada táctil como la salida gráfica del sistema.

Configuración del entorno y consideraciones prácticas

Durante el desarrollo, fue necesario ajustar manualmente los parámetros de calibración del panel táctil, identificando los valores mínimos y máximos para los ejes X e Y que correspondieran con las zonas útiles de la pantalla. Estos valores se introdujeron en el código fuente como constantes tras realizar pruebas directas y visualizar las coordenadas en el monitor serie. [18]

También se implementó una estructura de programa que permite conmutar entre modos de visualización y actualizar dinámicamente la pantalla sin interferencias visuales ni parpadeos, optimizando el uso de memoria RAM y evitando llamadas innecesarias a funciones de redibujado completo.

Por último, se organizó la estructura del proyecto en el entorno de desarrollo con ficheros comentados, separación de funciones según tareas (medición, visualización, interacción) y definición clara de variables globales para facilitar la modificación futura del código.

3.4. Programación de la placa arduino

El microcontrolador ha sido programado en lenguaje C++ utilizando el entorno Arduino IDE (código completo en A.1). El *firmware* implementa dos modos de funcionamiento seleccionables mediante un botón táctil en pantalla:

- **Modo Datos Numéricos:** muestra en tiempo real los valores de tensión (V) y potencia estimada (W) captados por la placa solar.

- **Modo Gráfica:** despliega una gráfica temporal del valor de tensión medida durante los últimos 20 segundos, permitiendo observar tendencias de captación solar.

El código realiza una lectura analógica continua del pin correspondiente a la señal de la placa solar, con un filtro de media móvil para suavizar la señal. La interfaz gráfica ha sido diseñada utilizando las librerías `Adafruit_GFX` y `MCUFRIEND_kbv`, y se ha implementado una navegación táctil básica entre menús.

Figura 3.3. Modo de visualización de datos en pantalla TFT

Figura 3.4. Modo de visualización de gráfica de potencia en tiempo real

Periodo de muestreo

Para la representación en tiempo real del nivel de tensión en la pantalla TFT, se ha definido un periodo de muestreo de 1000 ms (1 s). Este valor se ha seleccionado como compromiso entre una actualización suficientemente fluida de la gráfica y la capacidad de procesamiento del microcontrolador.

Un muestreo más rápido (por ejemplo, 100 ms) provocaría una carga excesiva en la interfaz gráfica, resultando en una actualización errática o incompleta de los datos en pantalla. Por otro lado, un muestreo demasiado lento (por encima de 5 s) generaría una pérdida de resolución temporal, dificultando la visualización de variaciones rápidas en la señal.

Este periodo de muestreo ha sido calibrado empíricamente durante las pruebas del sistema, permitiendo representar en pantalla una ventana móvil de 20 segundos de historial, con 20 muestras distribuidas uniformemente, lo que proporciona un equilibrio adecuado entre rendimiento gráfico y utilidad informativa.

3.5. Interfaz táctil y control de menús

Para el control del interfaz táctil, se ha realizado una calibración precisa de los ejes X-Y en la pantalla resistiva. Se han definido áreas interactivas en la zona inferior de la pantalla que permiten conmutar entre modos, así como volver al menú principal.

Se ha incorporado una lógica antirrebote y un retardo tras pulsación para evitar activaciones no deseadas. Esta parte del firmware asegura una experiencia de usuario estable y una respuesta táctil precisa.

3.6. Cableado, alimentación y ensamblaje final

Todos los elementos han sido conectados mediante cables tipo Dupont con conectores macho-hembra. La alimentación se realiza mediante un conversor de 5 V estabilizado, que suministra corriente tanto al Arduino como a la pantalla.

Finalmente, se ensambló toda la estructura, incluyendo la fijación de la placa solar en la cúspide de la torre, y se realizaron pruebas de funcionamiento estático, comprobando la visualización de datos y la correcta conmutación de modos en pantalla.

Figura 3.5. Vista final del montaje completo de la torre solar

Capítulo 4

Desarrollo del Heliostato Automatizado

4.1. Introducción

Este capítulo recoge el proceso completo de diseño, construcción y puesta en marcha del sistema de seguimiento solar, cuyo objetivo es orientar de forma automática un espejo (heliostato) de manera que refleje la radiación solar hacia el receptor situado en lo alto de la torre. A diferencia de sistemas clásicos de seguimiento astronómico mediante coordenadas solares predefinidas, se ha optado por una solución basada en visión artificial e IA, lo que permite una mayor adaptabilidad ante condiciones variables y reduce la necesidad de calibración previa.

4.2. Diseño del heliostato y soporte motorizado

La estructura del heliostato ha sido diseñada en SolidWorks considerando dos grados de libertad: eje horizontal (azimut) y eje vertical (elevación). El diseño incorpora una base rígida de soporte, sobre la cual se sitúa una columna vertical que permite la rotación en el eje horizontal (azimut). En la parte superior de esta columna está montado un mecanismo pivotante para el eje vertical (elevación), que sostiene la plataforma móvil del espejo. Todo el conjunto ha sido fabricado en impresión 3D utilizando PLA, con refuerzos estructurales en zonas críticas.

El sistema se ha montado sobre una base firme para evitar vibraciones, y se ha considerado la accesibilidad al cableado y motores durante la fase de montaje y pruebas.

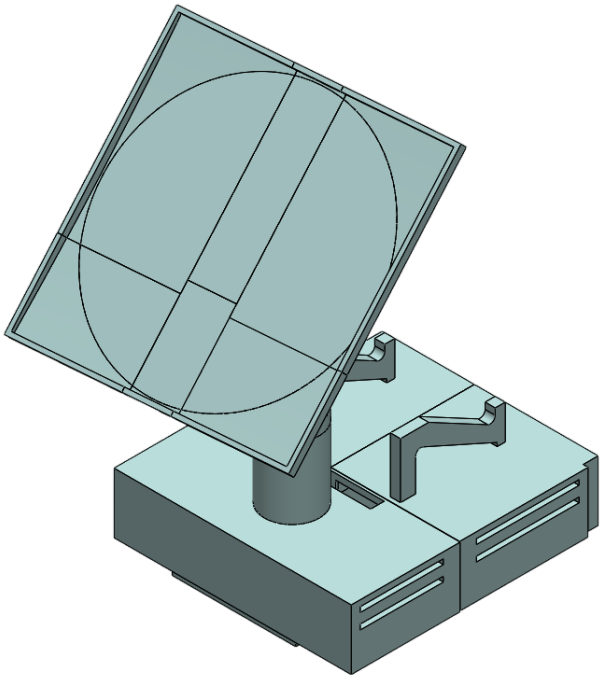

Figura 4.1. Diseño 3D del heliostato

4.3. Librerías y configuración del entorno para el heliostato

El heliostato se ha desarrollado sobre una Raspberry Pi 4, que se encarga tanto de la detección de objetos mediante visión artificial como del control de motores y la interfaz gráfica de usuario. Para ello, ha sido necesario configurar el sistema operativo, instalar librerías especializadas en Python y organizar el entorno de archivos y ejecución (código completo en A.2).

El sistema operativo utilizado ha sido Raspberry Pi OS Lite de 64 bits. En primer lugar, se activó la interfaz de la cámara CSI y se instalaron los paquetes básicos necesarios para trabajar con Python, librerías científicas, interfaces gráficas y control de hardware.

Para la captura de vídeo en tiempo real desde la cámara, se utilizó una herramienta de streaming que permite generar un flujo continuo en formato MJPEG. Este flujo es tratado directamente en Python mediante OpenCV para ser procesado por el sistema de visión [19].

Las principales librerías utilizadas en el sistema son:

- **opencv-python (cv2):** utilizada para la captura, preprocesamiento y visualización de imágenes.
- **tflite_runtime:** librería de TensorFlow Lite empleada para cargar y ejecutar el modelo

de detección optimizado.

- **pycoral:** utilizada para realizar inferencia con el acelerador Coral USB Edge TPU.
- **RPi.GPIO:** necesaria para controlar los pines GPIO que activan los motores a través del puente H L298N.
- **tkinter:** permite generar una interfaz gráfica sencilla que contiene los botones de control manual.
- **threading:** permite ejecutar procesos en paralelo, como la inferencia y la interfaz gráfica, sin bloquear la ejecución general del sistema.
- **numpy:** utilizada para cálculos numéricos y manipulación de arrays durante el análisis de detecciones.

El modelo de inteligencia artificial utilizado fue entrenado por la PSA. Este modelo, optimizado para ejecutarse con Coral TPU, está diseñado para detectar varias clases de objetos entre ellos el foco de luz y el panel fotovoltaico situado en la torre. Junto con el modelo, se utilizó un archivo de etiquetas que define las clases reconocidas.

En cuanto a la organización del proyecto en la Raspberry Pi, se estructuraron los archivos en una carpeta principal que contiene el *script* de control del heliostato, el modelo entrenado, las etiquetas de las clases detectadas y un directorio auxiliar con funciones adicionales. También se generó un archivo temporal donde se almacena el flujo de vídeo en tiempo real.

Durante la configuración del sistema se aplicaron diversas optimizaciones para mejorar el rendimiento, entre ellas la reducción de la resolución de la cámara para disminuir la latencia del procesamiento, la eliminación de mensajes de consola innecesarios que ralentizan la ejecución, y la asignación de mayor prioridad al proceso principal del sistema para asegurar una respuesta rápida y constante.

Gracias a esta configuración, el sistema heliostático funciona de forma fluida, estable y eficiente, siendo capaz de detectar y reaccionar ante los cambios en la posición del sol y del receptor en tiempo real, todo ello con un consumo energético y computacional muy reducido.

4.4. Componentes electrónicos y arquitectura de alimentación

El heliostato emplea dos motores de corriente continua de 12 V y 2 RPM con reductora integrada, seleccionados por su capacidad de generar alto par a baja velocidad, lo que resulta ideal para movimientos precisos y controlados sin necesidad de sensores de posición o codificadores.

Ambos motores están conectados a una controladora de motores basada en el chip L298N, que permite el control de dirección y encendido/apagado a través de señales lógicas. Esta controladora se alimenta directamente por una fuente de alimentación externa estabilizada de 12 V DC. A su vez, los motores reciben alimentación a través de los terminales de salida del L298N, lo que permite independizar la sección de potencia del circuito de la sección de control lógico.

La Raspberry Pi 4 actúa como unidad central de control, enviando señales digitales desde sus GPIOs [20] hacia los pines de control de la L298N. La asignación de pines es la siguiente:

- **Eje X (azimut):** GPIO 19 (IN1) y GPIO 16 (IN2)
- **Eje Y (elevación):** GPIO 26 (IN1) y GPIO 20 (IN2)

Este diseño permite una arquitectura robusta y segura, ya que la Raspberry Pi no alimenta directamente ningún actuador, lo que evita sobrecargas y preserva su integridad. La separación física y eléctrica entre lógica y potencia es una práctica habitual en sistemas industriales y de control embebido.

Para alimentar la Raspberry Pi se utiliza una fuente independiente de 5 V y 3 A, garantizando así un funcionamiento estable del sistema de procesamiento y visión artificial.

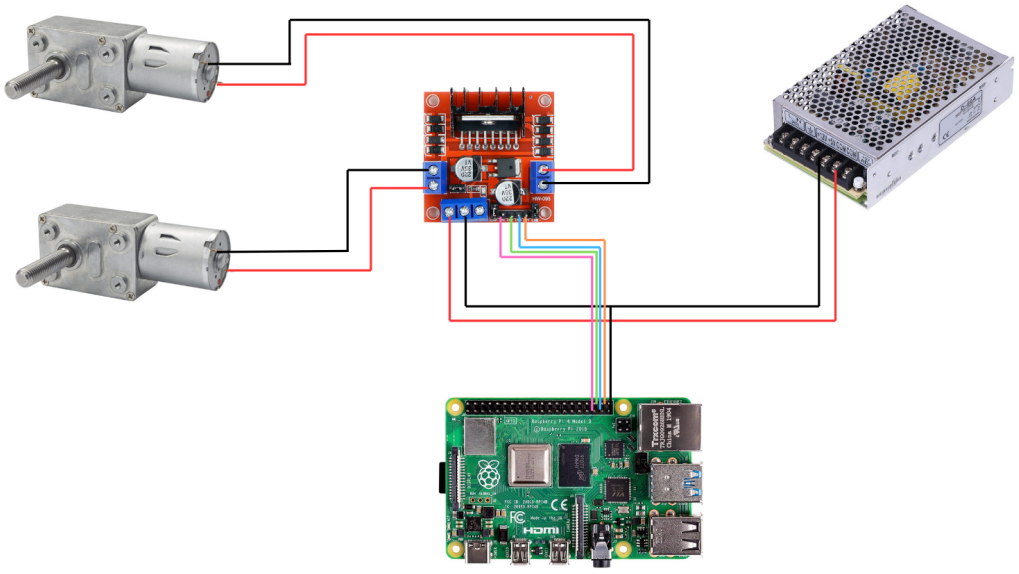

Figura 4.2. Esquema de control de motores mediante la controladora L298N conectada a Raspberry Pi

4.5. Sistema de visión artificial con modelo de la PSA

La detección visual es el núcleo funcional del seguimiento solar implementado. El sistema emplea una cámara oficial Raspberry Pi Camera V2.1 (sensor Sony IMX219), conectada directamente a la interfaz CSI de la Raspberry Pi. Esta cámara proporciona resolución de hasta 8 MP y una tasa de captura fluida (30 fps), suficiente para detección en exteriores y control en tiempo real.

El procesamiento de imagen se realiza mediante un modelo de inferencia entrenado específicamente por la PSA. Este modelo fue entrenado sobre imágenes reales de su propia instalación, permitiendo una detección robusta de dos clases esenciales:

- `SUN` — Región correspondiente al disco solar visible (foco de luz).
- `TARGET` — Superficie receptora situada en la torre (placa solar fotovoltaica).

Para entrenar el modelo de la PSA [21] se consideró una red neuronal artificial (ANN) preentrenada: SSD MobileNet V1.

Las MobileNets [22] son modelos de redes neuronales convolucionales (CNN) de código abierto diseñados para una visión eficiente en dispositivos. El detector de objetos utilizado por la ANN es el Single Shot Multibox Detector (SSD) [23]. Esta ANN fue entrenada inicialmente con un conjunto de imágenes del dataset Microsoft Common Objects in Context (COCO) [24].

Se utilizó una ANN preentrenada porque ya ha aprendido un conjunto rico de características generales que son útiles para nuevas redes neuronales. Esto se conoce como aprendizaje por transferencia (*transfer learning*), por lo tanto, la ANN se reentrena en lugar de entrenarse desde cero. El aprendizaje por transferencia también reduce el tiempo necesario para el nuevo proceso de entrenamiento.

La inferencia se lleva a cabo utilizando un Coral USB Accelerator con TPU (*Tensor Processing Unit*) [25], lo que descarga la unidad central de procesamiento (CPU) de la Raspberry Pi y permite tiempos de respuesta inferiores a 50 ms por imagen. El modelo, en formato TensorFlow Lite, se integra con la librería `tflite_runtime` y el flujo de vídeo se recibe desde un FIFO generado por el siguiente comando:

```
libcamera-vid --codec mjpeg -t 0 --width 640 --height 480 -o /tmp/cam_fifo.mjpg
```

El sistema analiza fotogramas en tiempo real, identifica los objetos de interés y calcula sus centroides para su posterior tratamiento en el algoritmo de control. La precisión del modelo ha sido validada en diferentes condiciones de iluminación con un alto nivel de acierto y estabilidad en la clasificación.

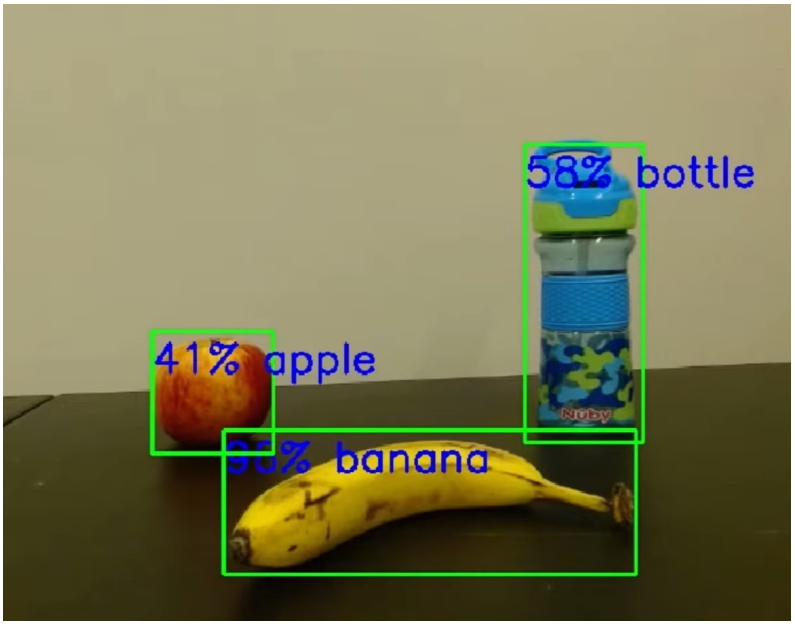

Figura 4.3. Detección de clases mediante inferencia con Coral TPU

4.6. Algoritmo de control y estrategia de seguimiento

El algoritmo de control del heliostato está basado en el principio de coincidencia entre el punto medio formado por el sol y el receptor, y el centro de la imagen captada por la cámara. Para cada *frame* procesado, se realiza el siguiente flujo lógico:

1. Detección de las clases `SUN` y `TARGET`, obtención de sus regiones (*bounding boxes*) y de sus coordenadas centrales $(x_{\text{sol}}, y_{\text{sol}})$ y $(x_{\text{torre}}, y_{\text{torre}})$.
2. Cálculo del punto medio de la linea recta que une las coordenadas centrales de ambas regiones:
$$x_{\text{centro}} = \frac{x_{\text{sol}} + x_{\text{torre}}}{2}, \quad y_{\text{centro}} = \frac{y_{\text{sol}} + y_{\text{torre}}}{2}$$
3. Comparación con el centro de la imagen (x_0, y_0).
4. Si $|x_{\text{centro}} - x_0| > \delta_x$, activar motor horizontal.
5. Si $|y_{\text{centro}} - y_0| > \delta_y$, activar motor vertical.
6. Apagar motores al alcanzar el margen tolerado.

Con el objetivo de evitar un movimiento excesivo o innecesario del heliostato debido a pequeñas oscilaciones o errores en la detección, se han definido dos umbrales de tolerancia angular

denominados δ_x y δ_y. Estos valores establecen una "zona muerta" en la que se considera que el sistema ya está correctamente alineado, y por tanto no se realiza corrección alguna. El umbral para $\delta_x = 60$ pixeles y para $\delta_y = 45$ pixeles

Dado que la resolución de la imagen utilizada es de 640×480 píxeles, se ha considerado oportuno definir umbrales de corrección distintos para los ejes X e Y. Esto se debe a que la proporción horizontal de la imagen (640 píxeles) es mayor que la vertical (480 píxeles), lo que implica que una misma desviación en píxeles representa una variación angular menor en el eje X que en el eje Y. Por tanto, para mantener una sensibilidad similar en ambos ejes, se ha aplicado un umbral horizontal más amplio que el vertical, ajustando así la precisión del sistema de centrado al formato de la imagen capturada.

Se ha implementado un sistema de retardo mínimo (*delay* de 100 ms) entre cada corrección para evitar sobreoscilaciones y prolongar la vida útil de los motores. Además, se utiliza una pequeña histéresis para evitar el activado/desactivado constante en valores frontera.

Esta estrategia de control simple pero efectiva permite un alineamiento dinámico y continuo con el sol, sin necesidad de GPS ni relojes astronómicos, y se adapta automáticamente a cambios de iluminación y orientación.

4.7. Interfaz manual para mantenimiento y pruebas

Para facilitar el ajuste inicial y la validación del sistema, se ha desarrollado una interfaz gráfica en Python basada en la librería `tkinter`. Esta interfaz permite controlar manualmente los movimientos del heliostato mediante cuatro botones (X+, X–, Y+, Y–), ubicados en la esquina inferior derecha de la pantalla. Un botón adicional permite detener ambos motores en caso de emergencia o centrado manual.

Los botones están integrados visualmente en la ventana principal que muestra el flujo de vídeo en tiempo real, lo que permite una comprobación visual instantánea de los efectos de cada pulsación.

Durante las pruebas, esta interfaz ha sido especialmente útil para:

- Calibrar la posición inicial del espejo.
- Validar la dirección y sentido de rotación de cada motor.
- Realizar correcciones en tiempo real sin necesidad de apagar el sistema.
- Diagnosticar errores de detección o bloqueos mecánicos.

Figura 4.4. Interfaz de control manual con botones táctiles integrados en la visualización

4.8. Montaje del heliostato

El montaje del heliostato se llevó a cabo utilizando un prototipo ya diseñado e impreso previamente por la PSA. La estructura, conformada por piezas impresas en 3D, incluía una base fija, un soporte vertical para el eje de elevación (Y) y una plataforma giratoria destinada al posicionamiento del espejo. Mi intervención se centró en el ensamblaje de dichos componentes mecánicos y en la integración completa del sistema electrónico: instalación de los motores de 12 V y 2 RPM, conexionado a la controladora de doble puente H y alimentación desde la fuente de 12 V. Además, se organizó y aseguró el cableado del sistema, dejando el conjunto operativo y listo para pruebas funcionales.

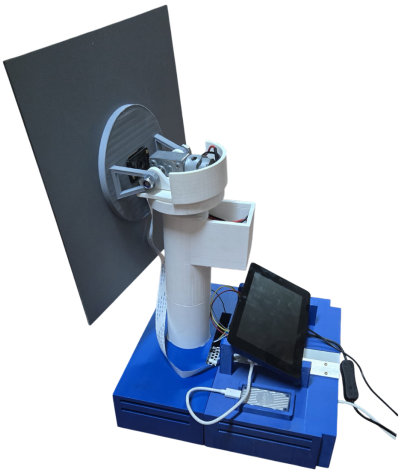

Figura 4.5. Prototipo del heliostato

4.9. Resultados de pruebas y validación del sistema

Una vez finalizado el montaje del heliostato y su integración con la Raspberry Pi, se realizaron diversas pruebas funcionales para verificar el correcto comportamiento del sistema de seguimiento solar. Las pruebas se llevaron a cabo en condiciones de iluminación natural con el objetivo de evaluar la respuesta del sistema en diferentes situaciones.

Durante las pruebas iniciales, el modelo de visión artificial fue capaz de detectar de forma fiable la posición del sol y del receptor simulado, activando los motores de forma coordinada para alinear el espejo con la torre. Se observó que, al superar los umbrales definidos, el sistema corregía su posición con un retardo medio inferior a 1 segundo, lo cual es aceptable para un sistema de baja potencia y propósito educativo.

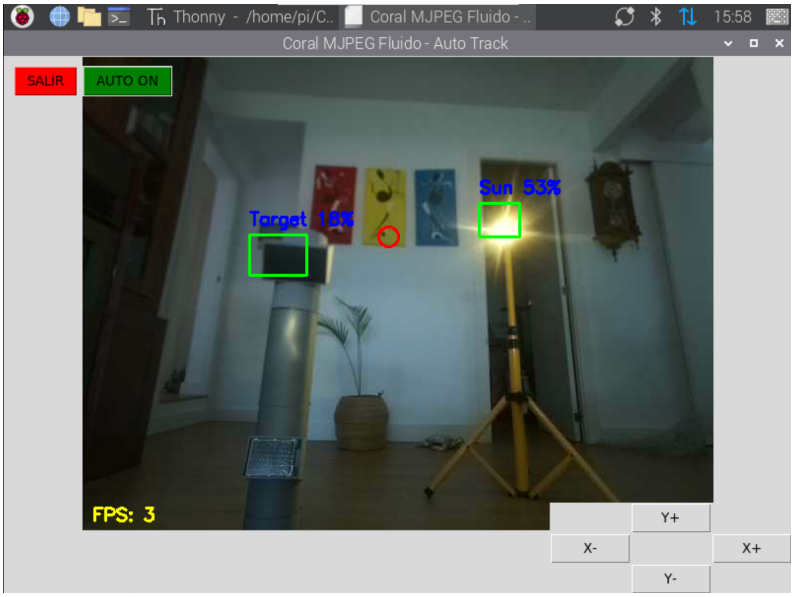

Figura 4.6. Detección del foco y la placa fotovoltaica

En condiciones de luz ambiental débil o fuentes de luz artificial intensa, el modelo mostró una ligera disminución en la estabilidad de detección. Esto sugiere que un entrenamiento más amplio del modelo podría mejorar la robustez general.

Para validar el correcto funcionamiento del sistema de visión artificial y seguimiento automático, se realizaron pruebas bajo condiciones de iluminación natural controlada. En un total de 10 ensayos, se evaluó la capacidad del sistema para identificar correctamente las entidades clave (foco y panel fotovoltaico) y centrar el espejo en consecuencia. Los resultados mostraron una tasa de acierto del 70 % en la detección del objeto "sol" y del 50 % en la detección del "*target*" o receptor. El sistema logró centrar ambos objetos en la mayoría de los casos, alcanzando un 50 % de éxito en el posicionamiento óptimo del espejo. Los fallos detectados se debieron

principalmente a condiciones de alto contraste que afectaban la inferencia del modelo Coral.

El movimiento de los motores resultó estable, con tiempos de reacción adecuados y sin comportamientos erráticos. La mecánica del soporte y los ejes del heliostato mantuvieron la orientación, lo que permitió mantener la reflexión hacia el receptor de forma constante durante varios minutos.

En general, los resultados confirman que el sistema cumple con los objetivos planteados, reproduciendo de forma realista el funcionamiento básico de un heliostato autónomo mediante visión artificial y control embebido.

Figura 4.7. Ensayos con el prototipo

Capítulo 5

Conclusiones y trabajos futuros

5.1. Conclusiones

El presente proyecto ha permitido desarrollar un prototipo funcional a escala de una planta solar de concentración de torre con receptor central, integrando tanto la torre con receptor simulado, como el heliostato inteligente capaz de orientar el espejo mediante control automático basado en IA y visión artificial.

A nivel de diseño mecánico y fabricación aditiva, se ha conseguido replicar la estructura principal de la torre mediante modelado CAD y FDM en impresoras 3D, garantizando robustez y funcionalidad que facilita la integración de componentes electrónicos y sensores.

En el ámbito electrónico, la utilización del microcontrolador Arduino Mega para la gestión de la torre ha permitido implementar con éxito una interfaz gráfica táctil que visualiza en tiempo real la potencia simulada recibida por el receptor, incluyendo modos de visualización numéricos y gráficos. La programación y el manejo eficiente de las librerías TFT aseguran una interacción fluida y una actualización constante de los datos adquiridos.

El desarrollo del heliostato ha sido un desafío técnico significativo, logrando integrar una Raspberry Pi 4 con un sistema de visión artificial basado en modelos entrenados por la PSA, combinado con una arquitectura de control de motores eléctricos mediante una controladora L298N. Este enfoque ha permitido implementar un seguimiento automático del sol con umbrales de tolerancia definidos, que optimizan la precisión del reflejo sin sobrecargar el sistema con movimientos innecesarios.

Las pruebas realizadas han validado el correcto funcionamiento del sistema completo, mostrando que el heliostato es capaz de ajustar su orientación en tiempo real para reflejar la radiación solar hacia la placa receptora en la torre, reproduciendo el principio básico de concentración solar en plantas reales. La comunicación efectiva entre los distintos dispositivos y la gestión integrada

de hardware y software demuestran la viabilidad técnica del prototipo.

La maqueta desarrollada formará parte de una actividad de divulgación científica organizada por la PSA, que ha sido aprobada dentro del programa oficial de la Noche Europea de los Investigadores 2025. Esta iniciativa, promovida por la Comisión Europea en el marco de las Acciones Marie Skłodowska-Curie, se celebra anualmente en más de 300 ciudades europeas con el objetivo de acercar la ciencia a la ciudadanía de una forma participativa, lúdica e interactiva. En el caso de Almería, el evento tendrá lugar el viernes 26 de septiembre de 2025. El estudiante participará activamente en esta jornada, presentando la maqueta al público asistente y contribuyendo a la labor de divulgación científica llevada a cabo por la PSA.

En resumen, el proyecto ha cumplido con sus objetivos planteados, combinando conceptos multidisciplinares de electrónica, informática, fabricación aditiva y control automático para simular de forma didáctica y funcional una planta solar de concentración con receptor central. Esta experiencia ha aportado un conocimiento profundo en el diseño y desarrollo de sistemas embebidos con aplicaciones en energías renovables, posicionando una base sólida para futuras ampliaciones y mejoras.

5.2. Trabajos futuros

Aunque el proyecto ha cumplido los objetivos iniciales, queda mucho margen para mejorar y profundizar en diversas áreas. En primer lugar, una mejora clara sería optimizar el modelo de detección basado en inteligencia artificial. Actualmente, aunque detecta adecuadamente el sol y el receptor, en ciertas condiciones, la precisión se ve comprometida. Por tanto, sería muy útil entrenar un modelo más robusto con un conjunto de datos más amplio y variado, mejorando la estabilidad y precisión del seguimiento solar. Se plantea la evaluación de nuevas arquitecturas de redes neuronales para la detección de objetos, con el objetivo de aumentar la precisión y reducir la latencia en la inferencia. En particular, podrían explorarse modelos más eficientes como *EfficientDet*, así como redes específicas para tareas de seguimiento de objetos (*object tracking*), lo cual permitiría una gestión más robusta del movimiento solar continuo o de múltiples focos simultáneos.

El entorno Coral TPU soporta diversos modelos optimizados para visión artificial en tiempo real [26], entre los que se encuentran alternativas al modelo SSD MobileNet utilizado en este trabajo.

En cuanto al prototipo físico del heliostato, también sería interesante diseñar y construir una versión más robusta y precisa mecánicamente, ya que se partía de un prediseño. Esto podría implicar construir una estructura más rígida que minimice vibraciones y holguras, optimizando y minimizando el uso de material 3D . Estas mejoras permitirían aumentar significativamente la precisión y repetibilidad del sistema, acercándolo más a un dispositivo real operativo en campo.

Finalmente, avanzar hacia una interfaz de usuario más completa, incluyendo calibración automática, reportes y monitoreo remoto inalámbrico, control manual desde dispositivos USB o Bluetooth, haría más fácil y versátil la operación del prototipo. En definitiva, el proyecto proporciona una base sólida sobre la cual desarrollar futuros trabajos que enriquezcan el aprendizaje y potencien la aplicabilidad técnica de este sistema de concentración solar.

Anexo A

Anexo

A.1. Código torre solar

Programa para torre solar desarrollado en Arduino, utilizando C++ como lenguaje base. Implementa una interfaz gráfica táctil para entrada de datos y visualización. El código se ejecuta en una placa compatible con pantallas TFT y utiliza las librerías Adafruit, MCUFRIEND_kbv y TouchScreen para manejar gráficos y entrada táctil.

```
#include <Adafruit_GFX.h>
#include <Adafruit_TFTLCD.h>
#include <MCUFRIEND_kbv.h>
#include <TouchScreen.h>

#define PIN_SOLAR A15

#define BLACK   0x0000
#define WHITE   0xFFFF
#define RED     0xF800
#define YELLOW  0xFFE0
#define BLUE    0x001F
#define GREEN   0x07E0

MCUFRIEND_kbv tft;

#define YP A3
#define XM A2
#define YM 7
#define XP 6
TouchScreen ts = TouchScreen(XP, YP, XM, YM, 300);
```

```
bool enMenuGrafica = false;
float tensionValores[20];
unsigned long tiempoInicio = 0;
unsigned long tiempoVoltaje = 0;
float voltajeAnterior = -1, potenciaAnterior = -1;
bool estadoSolLuna = false;

void setup() {
  Serial.begin(9600);
  tft.begin(tft.readID());
  tft.setRotation(1);
  tft.fillScreen(BLACK);
  dibujarMenuPrincipal();
}

void loop() {
  leerPantallaTactil();

  if (!enMenuGrafica) {
    if (millis() - tiempoVoltaje > 1000) {
      tiempoVoltaje = millis();
      actualizarValores();
    }
  } else {
    if (millis() - tiempoInicio > 1000) {
      tiempoInicio = millis();
      float voltaje = analogRead(PIN_SOLAR) * (5.0 / 1023.0);
      actualizarGrafica(voltaje);
    }
  }
}

void actualizarValores() {
  float voltaje = analogRead(PIN_SOLAR) * (5.0 / 1023.0);
  float potencia = voltaje * 0.2;

  if (voltaje != voltajeAnterior) {
    tft.fillRect(140, 40, 80, 20, BLACK);
    tft.setTextColor(WHITE);
    tft.setCursor(140, 40);
    tft.print(voltaje, 2);
    tft.print(" V");
    voltajeAnterior = voltaje;

    if (voltaje > 2.0 && !estadoSolLuna) {
```

```
      dibujarSol();
      estadoSolLuna = true;
    } else if (voltaje <= 2.0 && estadoSolLuna) {
      borrarRayosSol();
      dibujarLuna();
      estadoSolLuna = false;
    }
  }

  if (potencia != potenciaAnterior) {
    tft.fillRect(140, 120, 80, 20, BLACK);
    tft.setTextColor(WHITE);
    tft.setCursor(140, 120);
    tft.print(potencia, 2);
    tft.print(" W");
    potenciaAnterior = potencia;
  }
}

void dibujarMenuPrincipal() {
  tft.fillScreen(BLACK);
  tft.setTextSize(2);
  tft.setTextColor(WHITE);
  tft.setCursor(20, 40);
  tft.print("Tension:");
  tft.setCursor(20, 120);
  tft.print("Potencia:");
  dibujarBoton();

  float voltaje = analogRead(PIN_SOLAR) * (5.0 / 1023.0);
  if (voltaje > 2.0) {
    dibujarSol();
    estadoSolLuna = true;
  } else {
    dibujarLuna();
    estadoSolLuna = false;
  }
}

void dibujarBoton() {
  tft.fillRect(80, 200, 160, 40, GREEN);
  tft.setTextSize(2);
  tft.setTextColor(BLACK);
  tft.setCursor(100, 215);
  tft.print("Ver Grafica");
```

```
}

void leerPantallaTactil() {
  TSPoint p = ts.getPoint();

  pinMode(XM, OUTPUT);
  pinMode(YP, OUTPUT);

  if (p.z > 300) {
    int x = map(p.x, 120, 900, 0, 480);
    int y = map(p.y, 100, 950, 0, 320);

    if (!enMenuGrafica && x > 500 && x < 600 && y > 150 && y < 190) {
      enMenuGrafica = true;
      tft.fillScreen(BLACK);
      dibujarGrafica();
      tiempoInicio = millis();
      delay(300);
      while (ts.getPoint().z > 300);
      return;
    }

    if (enMenuGrafica && x > 500 && x < 600 && y > 150 && y < 190) {
      enMenuGrafica = false;
      tft.fillScreen(BLACK);
      dibujarMenuPrincipal();
      delay(300);

      while (ts.getPoint().z > 300);
      return;
    }
  }
}

void dibujarSol() {
  int x = 270, y = 80, r = 25;
  tft.fillCircle(x, y, r, YELLOW);
  tft.drawLine(x, y - 30, x, y - 45, YELLOW);
  tft.drawLine(x, y + 30, x, y + 45, YELLOW);
  tft.drawLine(x - 30, y, x - 45, y, YELLOW);
  tft.drawLine(x + 30, y, x + 45, y, YELLOW);
  tft.drawLine(x - 20, y - 20, x - 35, y - 35, YELLOW);
  tft.drawLine(x + 20, y - 20, x + 35, y - 35, YELLOW);
  tft.drawLine(x - 20, y + 20, x - 35, y + 35, YELLOW);
  tft.drawLine(x + 20, y + 20, x + 35, y + 35, YELLOW);
```

```
}

void borrarRayosSol() {
  int x = 270, y = 80, r = 25;
  tft.fillCircle(x, y, r + 10, BLACK);
  tft.drawLine(x, y - 30, x, y - 45, BLACK);
  tft.drawLine(x, y + 30, x, y + 45, BLACK);
  tft.drawLine(x - 30, y, x - 45, y, BLACK);
  tft.drawLine(x + 30, y, x + 45, y, BLACK);
  tft.drawLine(x - 20, y - 20, x - 35, y - 35, BLACK);
  tft.drawLine(x + 20, y - 20, x + 35, y - 35, BLACK);
  tft.drawLine(x - 20, y + 20, x - 35, y + 35, BLACK);
  tft.drawLine(x + 20, y + 20, x + 35, y + 35, BLACK);
}

void dibujarLuna() {
  tft.fillCircle(270, 80, 25, BLUE);
  tft.fillCircle(281, 78, 18, BLACK);
}

void dibujarGrafica() {
  tft.fillScreen(BLACK);
  tft.setTextColor(WHITE);
  tft.setTextSize(2);
  tft.setCursor(10, 10);
  tft.print("Grafica Potencia");
  tft.drawLine(30, 180, 300, 180, WHITE);
  tft.drawLine(30, 50, 30, 180, WHITE);
  tft.fillRect(80, 200, 160, 40, RED);
  tft.setTextSize(2);
  tft.setTextColor(WHITE);
  tft.setCursor(100, 215);
  tft.print("Volver");

  for (int i = 0; i < 20; i++) {
    tensionValores[i] = 180;
  }
}

void actualizarGrafica(float nuevoValor) {
  int valorEscalado = map(nuevoValor * 100, 0, 500, 180, 50);

  for (int i = 0; i < 19; i++) {
    tensionValores[i] = tensionValores[i + 1];
  }
```

```
  tensionValores[19] = valorEscalado;

  tft.fillRect(31, 51, 269, 128, BLACK);

  for (int i = 0; i < 19; i++) {
    int x1 = 30 + (i * 13);
    int y1 = constrain(tensionValores[i], 50, 180);
    int x2 = 30 + ((i + 1) * 13);
    int y2 = constrain(tensionValores[i + 1], 50, 180);
    tft.drawLine(x1, y1, x2, y2, YELLOW);
  }
}
```

Listing A.1. Código de la torre solar

A.2. Código heliostato

Programa para heliostato inteligente desarrollado en Python, ejecutado en una Raspberry Pi con Raspberry Pi OS Lite de 64 bits. Implementa una interfaz gráfica para seguimiento automático de objetos con una cámara y Coral USB Accelerator con TPU, usando detección por redes neuronales y control de motores.

```
# -*- coding: utf-8 -*-
import cv2
import numpy as np
import tkinter as tk
from PIL import Image, ImageTk
from tflite_runtime.interpreter import Interpreter, load_delegate
import threading
import subprocess
import os
import RPi.GPIO as GPIO
import time

FIFO_PATH = "/tmp/cam_fifo.mjpg"
WIDTH, HEIGHT = 640, 480

# Pines motores
MOTOR_X_IN1, MOTOR_X_IN2 = 19, 16
MOTOR_Y_IN1, MOTOR_Y_IN2 = 26, 20

GPIO.setmode(GPIO.BCM)
GPIO.setwarnings(False)
GPIO.setup(MOTOR_X_IN1, GPIO.OUT)
```

```
GPIO.setup(MOTOR_X_IN2, GPIO.OUT)
GPIO.setup(MOTOR_Y_IN1, GPIO.OUT)
GPIO.setup(MOTOR_Y_IN2, GPIO.OUT)

def stop_motors():
    for pin in [MOTOR_X_IN1, MOTOR_X_IN2, MOTOR_Y_IN1, MOTOR_Y_IN2]:
        GPIO.output(pin, GPIO.LOW)

# Coral con fallback a CPU
try:
    interpreter = Interpreter(
        model_path="model.tflite",
        experimental_delegates=[load_delegate("libedgetpu.so.1")]
    )
    print("Coral activado")
except Exception:
    interpreter = Interpreter(model_path="model.tflite")
    print("Coral no detectado, usando CPU")

interpreter.allocate_tensors()
input_details = interpreter.get_input_details()
output_details = interpreter.get_output_details()
labels = [line.strip() for line in open("labels.txt")]

# Stream MJPEG con AWB y autofocus
def iniciar_stream():
    if os.path.exists(FIFO_PATH):
        os.remove(FIFO_PATH)
    os.mkfifo(FIFO_PATH)
    cmd = [
        "libcamera-vid", "-n",
        "--width", str(WIDTH), "--height", str(HEIGHT),
        "--framerate", "30",
        "--codec", "mjpeg", "--inline",
        "--awb", "auto",
        "--autofocus-mode", "continuous",
        "--output", FIFO_PATH, "-t", "0"
    ]
    return subprocess.Popen(cmd, stdout=subprocess.DEVNULL, stderr=
        subprocess.DEVNULL)

# Interfaz
root = tk.Tk()
root.title("Coral MJPEG Fluido - Auto Track")
root.geometry(f"{WIDTH}x{HEIGHT}")
```

```
canvas = tk.Label(root)
canvas.pack()

def mover_y_mas(): GPIO.output(MOTOR_Y_IN1, GPIO.HIGH); GPIO.output(
    MOTOR_Y_IN2, GPIO.LOW)
def mover_y_menos(): GPIO.output(MOTOR_Y_IN1, GPIO.LOW); GPIO.output(
    MOTOR_Y_IN2, GPIO.HIGH)
def mover_x_mas(): GPIO.output(MOTOR_X_IN1, GPIO.HIGH); GPIO.output(
    MOTOR_X_IN2, GPIO.LOW)
def mover_x_menos(): GPIO.output(MOTOR_X_IN1, GPIO.LOW); GPIO.output(
    MOTOR_X_IN2, GPIO.HIGH)

modo_auto = False
AUTO_TOLERANCIA = 30
AUTO_DELAY = 0.2

def mover_automaticamente(center_x, center_y):
    dx = center_x - WIDTH // 2
    dy = center_y - HEIGHT // 2

    if abs(dx) > AUTO_TOLERANCIA:
        mover_x_mas() if dx > 0 else mover_x_menos()
        time.sleep(AUTO_DELAY)
        stop_motors()

    if abs(dy) > AUTO_TOLERANCIA:
        mover_y_mas() if dy > 0 else mover_y_menos()
        time.sleep(AUTO_DELAY)
        stop_motors()

def toggle_auto():
    global modo_auto
    modo_auto = not modo_auto
    if modo_auto:
        boton_auto.config(bg="green", text="AUTO ON")
    else:
        boton_auto.config(bg="lightgray", text="AUTO OFF")
        stop_motors()

def salir():
    stop_motors()
    GPIO.cleanup()
    if os.path.exists(FIFO_PATH):
        os.remove(FIFO_PATH)
```

```
    stream_proc.terminate()
    root.destroy()

# Botones con presion prolongada
frame = tk.Frame(root)
frame.place(relx=1.0, rely=1.0, anchor='se')

boton_yp = tk.Button(frame, text="Y+", width=6)
boton_ym = tk.Button(frame, text="Y-", width=6)
boton_xp = tk.Button(frame, text="X+", width=6)
boton_xm = tk.Button(frame, text="X-", width=6)

boton_yp.grid(row=0, column=1)
boton_ym.grid(row=2, column=1)
boton_xp.grid(row=1, column=2)
boton_xm.grid(row=1, column=0)

boton_yp.bind("<ButtonPress-1>", lambda e: mover_y_mas())
boton_yp.bind("<ButtonRelease-1>", lambda e: stop_motors())

boton_ym.bind("<ButtonPress-1>", lambda e: mover_y_menos())
boton_ym.bind("<ButtonRelease-1>", lambda e: stop_motors())

boton_xp.bind("<ButtonPress-1>", lambda e: mover_x_mas())
boton_xp.bind("<ButtonRelease-1>", lambda e: stop_motors())

boton_xm.bind("<ButtonPress-1>", lambda e: mover_x_menos())
boton_xm.bind("<ButtonRelease-1>", lambda e: stop_motors())

tk.Button(root, text="SALIR", command=salir, bg="red").place(x=10, y=10)
boton_auto = tk.Button(root, text="AUTO OFF", command=toggle_auto, bg="
    lightgray")
boton_auto.place(x=80, y=10)

latest_frame = None
lock = threading.Lock()

def frame_reader():
    global latest_frame
    cap = cv2.VideoCapture(FIFO_PATH, cv2.CAP_FFMPEG)
    cap.set(cv2.CAP_PROP_BUFFERSIZE, 1)
    while True:
        ret, frame = cap.read()
        if ret:
            with lock:
```

```
                latest_frame = frame.copy()

smoothed_boxes = {}
last_time = time.time()
frame_count = 0
FPS = 0

def smooth_box(label, new_box, alpha=0.5):
    if label not in smoothed_boxes:
        smoothed_boxes[label] = new_box
    else:
        old_box = smoothed_boxes[label]
        smoothed_boxes[label] = [int(alpha * n + (1 - alpha) * o) for n,
            o in zip(new_box, old_box)]
    return smoothed_boxes[label]

def actualizar_frame():
    global FPS, frame_count, last_time
    with lock:
        frame = latest_frame.copy() if latest_frame is not None else
           None

    if frame is not None:
        frame_rgb = cv2.cvtColor(frame, cv2.COLOR_BGR2RGB)
        resized = cv2.resize(frame_rgb, (300, 300), interpolation=cv2.
           INTER_LINEAR)
        input_data = np.expand_dims(resized.astype(np.float32) / 255.0,
           axis=0)

        sun_center = None
        target_center = None

        try:
            interpreter.set_tensor(input_details[0]["index"], input_data
               )
            interpreter.invoke()
            boxes = interpreter.get_tensor(output_details[0]["index"])
               [0]
            classes = interpreter.get_tensor(output_details[1]["index"])
               [0]
            scores = interpreter.get_tensor(output_details[2]["index"])
               [0]
        except Exception as e:
            print("Error Coral:", e)
        else:
```

```
            for i in range(len(scores)):
                if scores[i] > 0.3:
                    ymin, xmin, ymax, xmax = boxes[i]
                    x1, y1 = int(xmin * WIDTH), int(ymin * HEIGHT)
                    x2, y2 = int(xmax * WIDTH), int(ymax * HEIGHT)
                    class_id = int(classes[i])
                    label_text = labels[class_id] if class_id < len(
                        labels) else f"ID {class_id}"
                    label = f"{label_text} {int(scores[i]*100)}%"
                    x1, y1, x2, y2 = smooth_box(label, [x1, y1, x2, y2])
                    cv2.rectangle(frame_rgb, (x1, y1), (x2, y2), (0,
                        255, 0), 2)
                    cv2.putText(frame_rgb, label, (x1, y1 - 10), cv2.
                        FONT_HERSHEY_SIMPLEX, 0.6, (0, 0, 255), 2)

                    cx = (x1 + x2) // 2
                    cy = (y1 + y2) // 2
                    if label_text.lower().startswith("sun"):
                        sun_center = (cx, cy)
                    elif label_text.lower().startswith("target"):
                        target_center = (cx, cy)

        if modo_auto and sun_center and target_center:
            mid_x = (sun_center[0] + target_center[0]) // 2
            mid_y = (sun_center[1] + target_center[1]) // 2
            cv2.circle(frame_rgb, (mid_x, mid_y), 10, (255, 0, 0), 2)
            mover_automaticamente(mid_x, mid_y)

        frame_count += 1
        now = time.time()
        if now - last_time >= 1.0:
            FPS = frame_count
            frame_count = 0
            last_time = now
        cv2.putText(frame_rgb, f"FPS: {FPS}", (10, HEIGHT - 10), cv2.
            FONT_HERSHEY_SIMPLEX, 0.6, (255, 255, 0), 2)

        img = Image.fromarray(frame_rgb)
        imgtk = ImageTk.PhotoImage(image=img)
        canvas.imgtk = imgtk
        canvas.configure(image=imgtk)

    root.after(10, actualizar_frame)

stream_proc = iniciar_stream()
```

```
threading.Thread(target=frame_reader, daemon=True).start()
actualizar_frame()
root.mainloop()
```

Listing A.2. Código del heliostato

Bibliografía

[1] Dassault Systèmes, "SOLIDWORKS - software de diseño CAD 3D." `https://www.solidworks.com/es`, 2024.

[2] A. H. Alami, A. G. Olabi, A. Mdallal, A. Rezk, A. Radwan, S. M. A. Rahman, S. K. Shah, and M. A. Abdelkareem, "Concentrating solar power (CSP) technologies: Status and analysis," *International Journal of Thermofluids*, vol. 18, p. 100340, 2023.

[3] J. A. Carballo, J. Bonilla, N. C. Cruz, J. Fernández-Reche, J. D. Álvarez, A. Avila-Marin, and M. Berenguel, "Reinforcement learning for heliostat aiming: Improving the performance of solar tower plants," *Applied Energy*, vol. 377, p. 124574, 2025.

[4] J. A. Carballo, J. Bonilla, L. Roca, and M. Berenguel, "New low-cost solar tracking system based on open source hardware for educational purposes," *Solar Energy*, vol. 174, pp. 826–836, 2018.

[5] J. A. Carballo, J. Bonilla, M. Berenguel, J. Fernández-Reche, and G. García, "New approach for solar tracking systems based on computer vision, low cost hardware and deep learning," *Renewable Energy*, vol. 133, pp. 1158–1166, 2019.

[6] J. A. Carballo, J. Bonilla, M. Berenguel, J. Fernández-Reche, and G. García, "Solar tower power mockup for the assessment of advanced control techniques," *Renewable Energy*, vol. 149, pp. 682–690, 2020.

[7] M. Abadi, A. Agarwal, P.Barham, E. Brevdo, Z. Chen, C. Cirto, G. S. Corrado, A. Davis, J. Dean, M. Devin, S. Ghemawat, I. Goodfellow, A. Harp, G. Irvin, M. Isard, Y. Jia, R. Jozefowicz, L. Kaiser, M. Kudlur, J. Levenberg, D. Mané, R. Monga, S. Moore, D. Murray, C. Olah, M. Schuster, J. Shlens, B. Steiner, I. Sutskever, K. Talwar, P. Tucker, V. Vanhoucke, V. Vasudevan, F. Viégas, O. Vinyals, P. Warden, M. Wattenberg, M. Wicke, Y. Yu and X. Zheng, "Tensorflow: Large-scale machine learning on heterogeneous systems." Software available from tensorflow.org, 2015. `https://www.tensorflow.org/`.

[8] Adafruit Industries, "Adafruit GFX graphics library." `https://github.com/adafruit/Adafruit-GFX-Library`, 2022. Available at: `https://github.com/adafruit/Adafruit-GFX-Library`.

[9] D. Prentice, "MCUFRIEND_kbv TFT display library." https://github.com/prenticedavid/MCUFRIEND_kbv, 2021. Available at: https://github.com/prenticedavid/MCUFRIEND_kbv.

[10] Adafruit Industries, "Touchscreen library for Arduino." https://github.com/adafruit/Adafruit_TouchScreen, 2022. Available at: https://github.com/adafruit/Adafruit_TouchScreen.

[11] LEANTEC, "Documentación técnica del módulo controlador de motores L298N." https://leantec.es/wp-content/uploads/2019/05/LEANTEC-Documentacion-L298N-Rojo.pdf, 2019. Accedido en abril de 2025.

[12] OpenCV.org, "OpenCV: Open source computer vision library." https://pypi.org/project/opencv-python/, 2023. Available at: https://opencv.org/.

[13] C. R. Harris, K. J. Millman, S. J. van der Walt, R. Gommers, P. Virtanen, D. Cournapeau, E. Wieser, J. Taylor, S. Berg, N. J. Smith, R. Kern, M. Picus, S. Hoyer, M. H. van Kerkwijk, M. Brett, A. Haldane, J. F. del Río, M. Wiebe, P. Peterson, P. Gérard-Marchant, K. Sheppard, T. Reddy, W. Weckesser, H. Abbasi, C. Gohlke, and T. E. Oliphant, "Array programming with NumPy," *Nature*, vol. 585, no. 7825, pp. 357–362, 2020.

[14] B. Croston, "RPi.GPIO — A Python library for controlling GPIO on the Raspberry Pi." https://pypi.org/project/RPi.GPIO/, 2022. Available at: https://pypi.org/project/RPi.GPIO/.

[15] P. S. Foundation, "Tkinter — Python interface to Tcl/Tk." https://docs.python.org/3/library/tkinter.html, 2023. Available at: https://docs.python.org/3/library/tkinter.html.

[16] T. L. Team, "Tensorflow lite runtime for edge devices." https://www.tensorflow.org/lite/guide/python, 2023. Available at: https://www.tensorflow.org/lite/guide/python.

[17] M. McRoberts, "Chapter 8: LCD displays," in *Beginning Arduino*, pp. 139–160, Apress, 2013.

[18] M. McRoberts, "Chapter 12: Touch screens," in *Beginning Arduino*, pp. 219–240, Apress, 2013.

[19] A. Pajankar, *Raspberry Pi Image Processing Programming: With NumPy, SciPy, Matplotlib, and OpenCV*. Berkeley, CA: Apress, 2022.

[20] D. Molloy, "Chapter 8: Interfacing to the Raspberry Pi input/output pins," in *Exploring Raspberry Pi: Interfacing to the Real World with Embedded Linux*, pp. 239–278, Wiley, 2016.

[21] J. A. Carballo, J. Bonilla, J. Fernández-Reche, and D. C. Alarcón-Padilla, "Hel-iot web server: A smart heliostat development platform," in *SolarPACES Conference Proceedings*, vol. 1, 2024.

[22] A. Howard, M. Zhu, B. Chen, D. Kalenichenko, W. Wang, T. Weyand, M. Andreetto, and H. Adam, "Mobilenets: Efficient convolutional neural networks for mobile vision applications," *CoRR*, vol. abs/1704.04861, 2017.

[23] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. Reed, C. Fu, and A. Berg, "SSD: Single shot multibox detector," *CoRR*, vol. abs/1512.02325, 2015.

[24] T.-Y. Lin, M. Maire, S. Belongie, L. Bourdev, R. Girshick, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. Zitnick, "Microsoft COCO: Common objects in context," *CoRR*, vol. abs/1405.0312, 2014.

[25] A. Ghosh, S. Arefeen Al Mahmud, T. I. R. Uday, and D. F. Farid, "Assistive technology for visually impaired using Tensor Flow object detection in Raspberry Pi and Coral USB accelerator," in *2020 IEEE Region 10 Symposium (TENSYMP)*, pp. 186–189, IEEE, 2020.

[26] M. Tan, R. Pang, and Q. V. Le, "EfficientDet: Scalable and efficient object detection," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 10781–10790, 2020.

UNIVERSIDAD DE ALMERÍA

Resumen/Abstract

En el presente trabajo se ha desarrollado un prototipo funcional que simula el comportamiento de una planta solar de concentración con tecnología de torre, tomando como referencia el modelo de funcionamiento de la Plataforma Solar de Almería (PSA). El proyecto ha sido concebido con un enfoque didáctico y técnico, combinando electrónica embebida, diseño mecánico, visión artificial y control automático, integrando todos los conocimientos adquiridos a lo largo del grado.

El prototipo se compone de dos bloques principales: la torre con un receptor y un heliostato. La torre, diseñada en SolidWorks e impresa en 3D, incorpora una placa solar fotovoltaica en su parte superior, cuya señal se mide y visualiza mediante un sistema basado en Arduino MEGA 2560 y una pantalla táctil TFT. Esta interfaz permite al usuario consultar en tiempo real la tensión generada y visualizar gráficamente su evolución, simulando la radiación captada en una planta real.

Por otro lado, se ha implementado un heliostato funcional controlado por una Raspberry Pi 4, encargado de reflejar la luz solar sobre el receptor. Para lograrlo, se ha empleado una cámara y un modelo de inteligencia artificial entrenado por la propia PSA, capaz de detectar el foco de luz (simulando el sol) y la ubicación del panel fotovoltaico (simulando el receptor en la torre). A partir de esa información, el sistema gobierna dos motores de corriente continua a través de una controladora L298N, ajustando en tiempo real la orientación del espejo.

El conjunto del sistema ha sido validado mediante pruebas funcionales en condiciones reales, verificando la correcta alineación del haz reflejado, la respuesta del sistema ante variaciones de luminosidad y la estabilidad del control durante períodos prolongados. Además, se han implementado controles manuales que permiten verificar el funcionamiento del heliostato y facilitar tareas de mantenimiento.

Este proyecto demuestra la viabilidad de replicar el funcionamiento básico de una planta termosolar a pequeña escala utilizando tecnologías accesibles. A su vez, representa una herramienta útil tanto para la divulgación científica y técnica como para el análisis experimental de conceptos clave en energía solar, inteligencia artificial, automatización industrial y visión artificial.

In this project, a functional model has been developed to simulate the behavior of a central tower solar power plant, taking as reference the operational model of the Plataforma Solar de Almería (PSA). The project has been designed with both educational and technical purposes in mind, combining embedded electronics, mechanical design, computer vision, and automatic control—integrating all the knowledge acquired throughout the engineering degree.

The prototype consists of two main components: a tower with a receiver, and a heliostat. The tower, designed in SolidWorks and 3D printed, incorporates a photovoltaic solar panel at the top. Its output signal is monitored and displayed using an Arduino MEGA 2560-based system and a TFT touchscreen. This interface allows users to consult the generated voltage in real time and view its evolution graphically, simulating the radiation received in a real solar facility

On the other hand, a functional heliostat has been implemented and controlled by a Raspberry Pi 4, responsible for reflecting sunlight toward the receiver. To achieve this, a camera and an artificial intelligence model—trained by the PSA itself—are used to detect both the sun's position and the location of the receiver on the tower. Based on this information, the system controls two DC motors via an L298N motor driver, dynamically adjusting the mirror's orientation in real time.

The complete system has been validated through functional tests under real conditions, verifying the correct alignment of the reflected beam, the system’s response to changes in light intensity, and the stability of the control loop during extended operation. In addition, manual controls have been implemented to allow verification of the heliostat's operation and to facilitate maintenance tasks.

This project demonstrates the feasibility of replicating the basic operation of a solar thermal power plant at small scale using accessible technologies. Furthermore, it serves as a valuable tool for scientific and technical outreach as well as for the experimental analysis of key concepts in solar energy, artificial intelligence, industrial automation, and artificial vision.