UNIVERSITAT POLITÈCNICA DE VALÈNCIA

Escuela Técnica Superior de Ingeniería del Diseño

DISEÑO Y PROGRAMACIÓN DE UN SISTEMA DE BAJO COSTE BASADO EN LA ESP 32-CAM PARA EL CONTROL DE UN BRAZO ROBÓTICO

Trabajo Fin de Grado

Grado en Ingeniería Electrónica Industrial y Automática

AUTOR/A: Martín Osuna, Juan José

Tutor/a: Armesto Ángel, Leopoldo

Cotutor/a: Rodas Jordá, Ángel

CURSO ACADÉMICO: 2021/2022

UNIVERSITAT POLITÈCNICA DE VALÈNCIA

Escuela Técnica Superior de Ingeniería del Diseño

UNIVERSITAT POLITÈCNICA DE VALÈNCIA

Escuela Técnica Superior de Ingeniería del Diseño

DISEÑO Y PROGRAMACIÓN DE UN SISTEMA DE BAJO COSTE BASADO EN LA ESP 32-CAM PARA EL CONTROL DE UN BRAZO ROBÓTICO

TRABAJO FINAL DEL

Grado en Ingeniería Electrónica Industrial y Automática

REALIZADO POR

Juan José Martín Osuna

TUTORIZADO POR

Leopoldo Armesto Ángel

Ángel Rodas Jordá

CURSO ACADÉMICO: 2021/2022

Resumen:

Este proyecto de final de grado consistirá en el desarrollo de un sistema de funcionamiento autónomo de un robot, para el almacenaje y clasificación de piezas. Estará fundamentado en visión artificial, usando una cámara de bajo coste de la familia Esp 32. la Esp32 - CAM. Programada mediante la biblioteca openCV en lenguaje Python. El funcionamiento y control del robot se hará mediante placa y código basada en Arduino e implementando un sistema de comunicación serie entre ambos periféricos. Adicionalmente se desarrollarán algoritmos propios de visión artificial para procesar la imagen de manera embebida en la cámara, mediante el procesador de la misma.

Palabras clave:

Arduino, OpenCV, Python, Esp32, visión artificial, Esp 32 - CAM, brazo robótico, detección de color, procesado de imagen, comunicación serie.

Abstract:

This final degree project consists in developing the autonomous operating system of a robot, to stock and classify pieces. It will be based on artificial vision, using a low cost camara from the Esp 32 family, the Esp 32-CAM, and being programated using OpenCV library of Python lenguage. The operation and control of the robot is going to be done with a board and code based on arduino, and implementing a serial communication system between both peripherals. In addition, own artificial vision algorythms will be vedeloped to process the image embedded in the camera, through its processor.

Keywords:

Arduino, OpenCV, Python, Esp32, Artificial vision, Esp 32 - CAM, Robotic arm, Color detection, Image processing, Serial communication.

Agradecimientos:

Con estas últimas palabras pongo fin no solo a uno de los trabajos más importantes de mi vida, sino a una de las etapas que más me han hecho crecer como persona, han pasado cuatro años desde que empezó mi experiencia como universitario, y aunque ya llevo cientos de trabajos a mis espaldas, es imposible que en especial con este no tenga esos mismo nervios del primer día.

Pero si algo tengo claro, es que hasta aquí no podría haber llegado yo solo, este trabajo es la suma de todas las personas que forman parte de mi vida y por tanto que me hacen ser como soy. Por eso dedico estas líneas para darle las gracias.

A mis padres, y en general a toda mi familia, por darme esta oportunidad, brindarme todas las facilidades que han podido y sobre todo confiar en mí, siempre han estado ahí y eso para mi lo es todo.

A mis amigos, tanto mis colegas del cole, que están hasta las narices de oír hablar del tfg, pero que aun así me han escuchado cada una de las veces. Mis amigos del pueblo, que del tiempo que hace que no voy no sé si se acordarán de mi cara y mis amigos de las cartas, con los que me he saltado más entrenes de los que estoy dispuesto a admitir. Gracias por entenderme y jamás enfadaros por no dedicaros el tiempo que os mereceis.

Por último, gracias a mi pareja, una persona maravillosa, que ha llegado a mi vida en esta etapa y ha sido mi mayor apoyo siempre, apoyándome en los momentos imposibles y celebrando las pequeñas victorias. Siempre juntos, como un equipo.

Gracias por todo, hoy y siempre…

Índice de contenido

MEMORIA TÉCNICA **12**

1.1. OBJETO DEL PROYECTO Y MOTIVACIÓN 14

1.2. ANTECEDENTES 15

1.2.1. Presente de la robótica industrial y su papel en la educación 15

1.2.2. Introducción a la histórica de la visión artificial 18

1.3. ESTADO DEL ARTE 19

1.3.1. Proyectos profesionales 19

1.3.2. Trabajos académicos 20

1.4. ANTECEDENTES DEL PROYECTO 20

1.4.1. Brazo robot paralelogramo 20

1.4.2. Código de movimiento el robot 21

1.5. MATERIAL Y MÉTODOS 22

1.5.1. Material de fabricación piezas de color 22

1.5.2. Cámara para la etapa de pruebas de viabilidad el proyecto 24

1.5.3. Cámara para la implementación final del proyecto 26

1.5.4. Material para la base de fijación 27

1.5.5. Lenguaje de programación 28

1.6. DESARROLLO DE LA SOLUCIÓN 29

1.6.1. Control no embebido mediante webcam 29

1.6.1.1. Posición de los elementos y fijación de la cámara 30

1.6.1.2. Programación en python para la detección de piezas 31

1.6.1.3. Programación en Arduino para la interpretación de datos y el desplazamiento del robot 35

1.6.2. Control no embebido mediante ESP 32-CAM 36

1.6.3. Control embebido mediante ESP 32-CAM 39

1.6.3.1. Programación de la etapa de obtención de imagen 40

1.6.3.2. Programación de la etapa de segmentación RGB 40

1.6.3.3. Programación de la etapa de conversión de RGB a HSV 41

1.6.3.4. Programación de la etapa de umbralizado 41

1.6.3.5. Programación de la etapa de labeling 42

1.6.3.6. Programación de la etapa de cálculo del centro de gravedad 43

1.6.3.7. Conexión serie de la ESP 32-CAM y la placa Arduino Nano 44

1.7. TRABAJO RESTANTE Y FUTUROS DESARROLLOS 45

1.8. CONCLUSIÓN 45

1.9. BIBLIOGRAFÍA 46

ANEXOS **48**

ANEXO A-1: Código Arduino de partida para el control del brazo robot 50

ANEXO A-2: Código Python para webcam 60

ANEXO A-3: Código Arduino para control del robot a través del monitor serie 66

ANEXO A-4: Código Python para ESP 32-CAM 78

ANEXO A-5: Código Arduino para el labeling de la imagen embebida en la ESP32-CAM 84

ANEXO A-6: Código Arduino para el cálculo del centro de gravedad de la imagen embebida en la ESP32-CAM 91

ANEXO A-7: Programación de la ESP 32-CAM mediante el IDE de arduino 98

PLANOS **103**
2.1. Plano Pieza de color 105
2.2. Plano Recipientes 106
2.3. Plano Base de fijación 107
2.4. Plano Soporte camaras 108
2.5. Plano de conjunto del robot paralelogramo 109

PLIEGO DE CONDICIONES **110**
3.1. Objeto 112
3.2. Normativa 112
3.3. Materiales 112
3.3.1. Características principales 112
3.3.2. Control de calidad 114
3.4. Condiciones de ejecución 114
3.5. Control de calidad 115

PRESUPUESTO **116**
4.1. Tablas de costes por naturaleza del elemento 118
4.2. Tablas resumen 121

Índice de Figuras

Figura 1. Esquema de etapas y objetivos del proyecto
Figura 2. Imágen robot industrial scara de la marca delta . [1]
Figura 3. Imágen robot industrial antropomórfico de la marca ABB . [2]
Figura 4. Imágen robot industrial cartesiano . [3]
Figura 5. Imágen robot industrial paralelo de la marca DR . [4]
Figura 6. Diagrama de bloques que representa el funcionamiento de un sistema de visión artificial para la detección de tapones defectuosos en botellas. [5]
Figura 7. Imagen real del robot en posición de reposo, con el que se ha realizado el proyecto
Figura 8. Imágen cubos de madera de tamaño y forma similar al deseado . [10]
Figura 9. Imágen cubos de Poliestireno expandido de tamaño y forma similar al deseado.[11]
Figura 10. Imágen webcam modelo hama c-400. [12]
Figura 11. Imágen cámara ESP 32 - CAM Camera. [13]
Figura 12. Imágen cámara Raspberry Pi Camera. [14]
Figura 13. Imágen plancha de aluminio de 4 mm de espesor. [15]
Figura 14. Imágen plancha de contrachapado de 4 mm de espesor. [16]
Figura 15. Esquema de etapas. muestra los componentes que forman cada una de ellos así como la relación entre los mismos
Figura 16. Imágen tomada de la prueba realizada para determinar el espacio de tarea.
Figura 17. Imágen tomada desde la webcam para determinar la posición idónea de la misma.
Figura 18. Diagrama de funcionamiento del programa de python para la detección de piezas
Figura 19. Fragmento del código utilizado para definir el puerto y la frecuencia de conexión de python con arduino a través del puerto serie .
Figura 20. Imagen de espectro de colores para HSV . [17]
Figura 21. Fragmento del código utilizado para definir los límites altos y bajos ,en HSV, de los colores a detectar.
Figura 22. Fragmento del código utilizado para definir las mascaras de color segun los limites definidos y posteriormente detección de los contornos en función de esos colores.
Figura 23. Imagen tomada desde la webcam que ilustra la detección de colores y la obtención de las coordenadas x e y de los centros de gravedad de las piezas .
Figura 24. Esquema ejemplo cadena de información enviada por el ordenador al arduino para el control del robot
Figura 25. Diagrama de funcionamiento del programa de Arduino para el control del movimiento del robot
Figura 26. Diagrama que muestra los movimientos realizados en la función dejar rojo.
Figura 27. Foto del sistema resultante de la etapa uno funcionando tras la integración de todos los elementos.
Figura 28. Diagrama de comunicación entre los componentes del sistema
Figura 29. Fragmento del código utilizado para leer de la url y buscar los bytes de interés.
Figura 30. Imagen tomada desde la ESp 32-CAM que ilustra la detección de colores y la obtención de las coordenadas x e y de los centros de gravedad de las piezas
Figura 31. Foto del sistema resultante de la etapa dos funcionando tras la integración y adaptación de todos los elementos.
Figura 32. Diagrama de las etapas de procesador de imagen
Figura 33. Fragmento del código utilizado para separar por aritmética de punteros las componentes r,g y b de cada píxel que compone la imagen .
Figura 34. Imagen tomada al monitor serie de arduino que representa el proceso de umbralizado de los pixeles de una imagen
Figura 35. Imagen representativa de un proceso de labeling en una imagen binaria [19]
Figura 36. Fragmento del código utilizado para recorrer la matriz umbralizada y calcular el centro de gravedad
Figura 37. Imagen real tomada por la ESP32-CAM, que permite hacer visible el problema de ruido estático que sufre la cámara.
Figura 38. Diagrama de conexión serie y alimentación entre Arduino y ESP 32-CAM

Figura 39. Foto del programador esp32 cam MB.
Figura 40. Foto del programador ft232rl.
Figura 41. Imagen de la configuración del IDE de Arduino para subir los programas a la ESP 32 - CAM
Figura 42. Imagen para la identificación del botón de reset en la ESP 32 - CAM
Figura 43. Imagen cenital tomada para referenciar la posición final de los elementos que constituyen el proyecto

UNIVERSITAT POLITÈCNICA DE VALÈNCIA

Escuela Técnica Superior de Ingeniería del Diseño

MEMORIA TÉCNICA

DISEÑO Y PROGRAMACIÓN DE UN SISTEMA DE BAJO COSTE BASADO EN LA ESP 32-CAM PARA EL CONTROL DE UN BRAZO ROBÓTICO

TRABAJO FINAL DEL

Grado en Ingeniería Electrónica Industrial y Automática

REALIZADO POR

Juan José Martín Osuna

TUTORIZADO POR

Leopoldo Armesto Ángel

Ángel Rodas Jordá

CURSO ACADÉMICO: 2021/2022

1.1. OBJETO DEL PROYECTO Y MOTIVACIÓN

El proyecto nace, tras la idea de desarrollar una actividad que pueda unificar dos trabajos de dos asignaturas que se imparten en el cuarto año del grado Ingeniería Electrónica Industrial y Automática, en la nueva mención de robótica.

Por un lado, la asignatura de sistemas Robotizados, la cual, se imparte para todas las menciones de la carrera y que cuenta ya con un trabajo basado en montar y programar un robot paralelogramo.

Por otro lado, la asignatura de robótica inteligente , que se impartira por primera vez en el año 2022/2023, busca una actividad basada en algoritomos basicos de visión artificial, como puedan ser algoritmos de detección de color.

Ante estas dos premisas, se propone el desarrollo del proyecto, que consistirá en, partiendo del robot paralelogramo realizado en la asignatura de sistemas robotizados, desarrollar, mediante visión artificial, un sistema que permita detectar la posición y los colores de unas piezas determinadas. Pudiendo, de esta forma, transmitir esa información al brazo, con el objetivo de realizar una tarea de pick & place mediante una cámara lo más económica posible, para que pueda ser replicada por los alumnos en la asignatura sin suponer un coste muy elevado.

Con estos antecedentes, nace mi interés por realizar este proyecto, ya que me permitirá no solo trabajar en dos de los campos, para mí, más interesantes de mi carrera como son la programación y la informática, sino, además, formarme en la rama de la visión artificial, una disciplina que siempre ha despertado mucho interés en mi. Por lo tanto, este proyecto aúna todo lo necesario para ser el trabajo ideal para mi proyecto de fin de Grado.

Como objetivo principal del proyecto, está la tarea de analizar y probar la viabilidad de hacer una versión funcional de la tarea de pick & place mediante la implementación de algoritmos de visión artificial, para, a continuación, trasladar esa lógica del tratamiento de imagen a algoritmos de bajo nivel para programar la ESP 32-CAM, con el objetivo de obtener una versión del sistema embebida, que aproveche el procesador de la cámara para realizar un procesamiento de imagen lo más completo posible para la tarea en cuestión, sin tener que depender del procesador de un ordenador.

Pero, debido a mi bajo conocimiento en la visión artificial, antes de llegar al desarrollo de la razón última del proyecto, será necesario realizar y detenerse en el desarrollo de etapas intermedias, que permitan, de una manera más sencilla, familiarizarse con la programación y los algoritmos de visión artificial y sobre todo ir afrontando de manera segmentada las distinta soluciones que conformarán el proyecto final

Por lo que respecta a las especificaciones de la tarea, el robot ha de ser capaz de distinguir entre elementos distintos. En este caso, se utilizará una distinción por colores, pero este mismo funcionamiento es fácilmente replicable para distinción entre otras características, como podrían ser formas o posiciones dentro de piezas iguales.

Deberá producirse una tarea de almacenaje, que consistirá en, una vez distinguido el color y determinada la pieza en el espacio de trabajo, ser capaz de recogerla y depositarla en un recipiente con posición previamente definida.

Adicionalmente, se ha de asegurar que los distintos elementos que forman el sistema estén fijos. De ahí, la necesidad de diseñar una base a la que fijar los elementos.

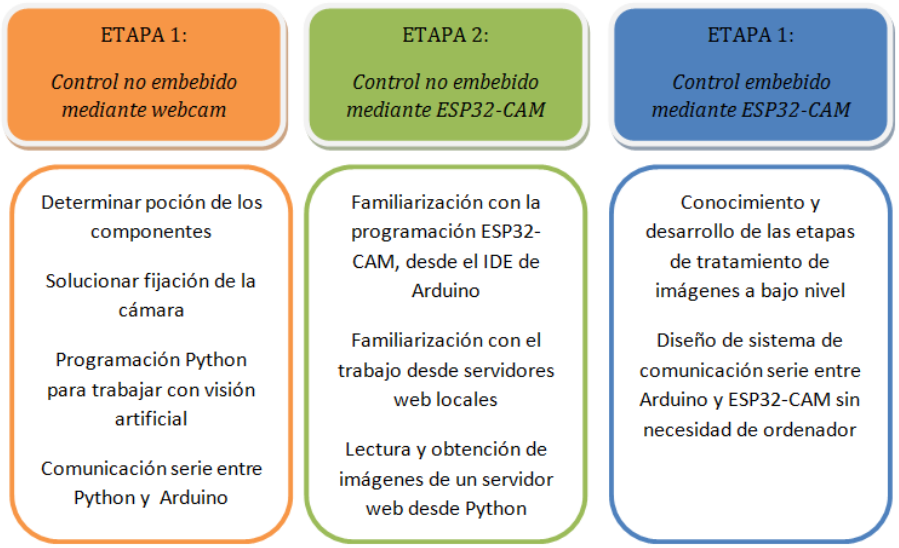


Figura 1. Esquema de etapas y objetivos del proyecto

De esta manera, y con los objetivos para cada etapa ya definidos (figura 1), se llega a la redacción de este proyecto, con una finalidad académica. Permitiendo durante su elaboración y en su posterior réplica como actividad académica, profundizar en los campos de la programación, la visión artificial y la robótica.

1.2. ANTECEDENTES

1.2.1. Presente de la robótica industrial y su papel en la educación

Con el nacimiento de la robótica hace más de 70 años, nace una de las ramas de desarrollo técnico industrial más importantes del mundo, ya que, desde bien pronto, la robótica ha pasado a jugar un papel fundamental en la industria. De ahí la existencia hoy en día de una gran pluralidad de robots distintos que permiten suplir actividades concretas dependiendo de las necesidades de las mismas.

Existen una gran variedad de modelos de robot, con sus características propias, pero la gran mayoría responden a modificaciones constructivas o variaciones de 4 grandes tipos de robots industriales. Estos son:

- **Robot Scara**, cuenta con una estructura cinemática abierta en serie. Dispone de 4 grados de libertad, siendo tres de estos para la posición en los ejes X, Y y Z y el último grado de libertad para la orientación del efector final.

Figura 2. Imágen robot industrial scara de la marca delta . [1]

Su principal aplicación en la industria suele ser para la inserción de componentes, debido a su alta precisión.

- **Robot Antropomórfico**, también cuenta con una cadena cinemática abierta en serie, su estructura se asemeja a un brazo humano, de ahí el hecho de contar con 6 grados de libertad, con el objetivo de poder desplazarse en todas las posiciones y disponer de todas las orientaciones en su efector final.

Figura 3. Imágen robot industrial antropomórfico de la marca ABB . [2]

Cabe destacar que, este tipo de robots, son los más utilizados en la industria. Además en muchas ocasiones se les añade una articulación adicional, es decir se les dota de 7 grados de libertad para hacerlos redundantes. De esta manera no solo puede alcanzar cualquier posición y orientación sino que lo pueden alcanzar con distintas posiciones, lo cual favorece su implementación junto a otros robots en el aspecto del espacio físico.

- **Robot cartesiano**: Es un tipo de robot muy particular, aunque con cadena cinemática abierta en serie, solo cuenta con tres grados de libertad, los cuales se emplean para su desplazamiento en los tres ejes.

Figura 4. Imágen robot industrial cartesiano . [3]

Su principal ventaja frente a los otros modelos es la sencillez de su cinemática y lo más importante su capacidad para desplazar objetos pesados, frente a los otros modelos.

- **Robot Paralelo:** este robot al contrario que los otros presenta una cadena cinemática cerrada en paralelo y al igual que el robot cartesiano cuenta con solo tres grados de libertad.

Figura 5. Imágen robot industrial paralelo de la marca DR . [4]

Este tipo de robots son los más utilizados para tareas de pick & place, debido a su alta velocidad de trabajo como consecuencia de la cadena de cinemática cerrada. Cabe destacar que el robot paralelogramo empleado en este proyecto se trata de un subtipo de estos robots.

Ante esta gran variedad de modelos, y sobre todo ante la gran importancia que han cobrado los robots industriales, es necesario contar con unos profesionales capaces de diseñar y trabajar con los distintos robots. En ese punto, entra en juego el papel de los robots de carácter educativo, que, de manera reducida y sencilla replicar los modelos reales para permitir a los estudiantes aprender y formarse en los ámbitos de la robótica.

Hoy en día, hay una gran diversidad de robots educativos, ajustados a todos los ámbitos, ya sea robots que ayudan a aprender los fundamentos de la programación, hasta réplicas idénticas de robots reales.

Algunos de los más importantes, son por ejemplo los kits de desarrollo de LEGO. La empresa de juguetes de construcción lleva desde los años 80 fusionando los bloques de construcción con componentes eléctricos, con el objetivo de no solo formarse en los conceptos constructivos de los robots sino aprender programación dentro de entornos sencillos y muy visuales.

Otro evento importante para la robótica educativa, es el papel que cumplen placas de desarrollo como arduino. Este tipo de productos, junto su gran variedad de modelos de servo permite la creación de una gran diversidad de modelos de robots,no solo por la flexibilidad de la placa, sino también por su popularidad, ya que es muy fácil encontrar proyectos de todo tipo que se ajusten a las necesidades de cualquier proyecto.

Además, la popularidad en los últimos años de equipos de impresión 3D de carácter doméstico ha dotado a estos robots con procesador Arduino de una herramienta muy versátil a la hora de la construcción de las estructuras para los robots.

Por otra parte, la variedad de sensores con las que cuenta Arduino permiten no solo la construcción del robot sino la implementación de herramientas sensoriales para un control más complejo, como podría ser la implementación de una cámara para una tarea de pick & place.

1.2.2. Introducción a la histórica de la visión artificial

La visión ha sido y es el sentido más importante para los seres humanos, de ahí, la importancia que se le ha dado siempre a intentar dotar de este mismo sentido a las máquinas y robots. Con el fin de lograr este objetivo nace la visión artificial.

Aunque, en primera instancia se pueda pensar que la visión artificial depende únicamente de la cámara que toma las fotos, esto no es así. La visión artificial, es una disciplina que procesa y analiza imágenes del mundo real, con el objetivo de convertir dichas imágenes en datos y procesarlas.

El papel de la cámara es la primera etapa de la visión artificial, por eso, el nacimiento de esta disciplina está estrechamente ligado con la invención de la primera cámara digital. Esta nace en 1975 de la mano de su inventor, Steve Sasson, para la compañía de fotografía Kodak.

Al final de los años 70, principios de los 80, con la revolución electrónica y, en consecuencia, con el auge de los microprocesadores y las cámaras de video CCD, nace la conocida como visión artificial. Aunque de manera muy primaria, esta mejora en los procesadores permitió captar, procesar y reproducir imágenes. Las infinitas posibilidades de esta nueva tecnología produjeron una rápida implementación en muchos sectores de carácter industrial, hasta convertirse, hoy en día, en una de las disciplinas más importantes de la informática y la electrónica.

Actualmente,la visión artificial es una realidad, y uno de los elementos vitales que permitieron llevar a la industria en algunos sectores a un punto de automatización prácticamente total. Uno de los puntos de más uso de esta tecnología, es en el control final de una cadena de montaje. Este punto, puede ser uno de los principales cuellos de botella en una producción lineal, ya que, hasta la implementación de la visión artificial, la comprobación de los productos estaba limitada por la velocidad de los operarios. Con la implementación de un sistema de visión y un actuador se puede realizar la comprobación y separación de manera dinámica, disminuyendo al mínimo posible los retrasos producidos por esta etapa. A continuación, se muestra un esquema resumen de la implementación de la visión artificial en un proceso industrial real (figura 6).

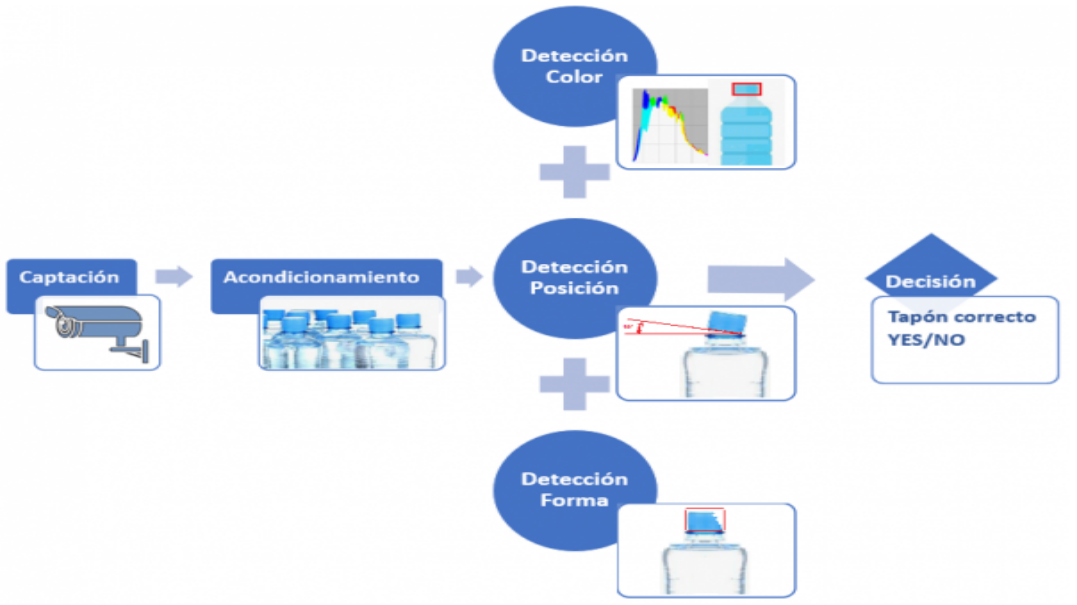


Figura 6. Diagrama de bloques que representa el funcionamiento de un sistema de visión artificial para la detección de tapones defectuosos en botellas. [5]

Aunque el potencial de la visión artificial en sí mismo ya es muy alto, la visión artificial también destaca por su capacidad de combinación con otras disciplinas. Entre otros, es remarcable el uso de la visión artificial para metrología, como puede ser el tamaño de una superficie o la distancia a un objeto;u uso para la lectura de códigos o caracteres, como pueden ser los códigos de barras o los códigos QR o incluso como sistema de control para robots, para actividades de pick & place.

Otro de los papeles que hace tan importante la visión artificial, es el papel que desempeña como complemento de la inteligencia artificial. Por una parte, la implementación de la visión artificial permite dotar a los sistemas de inteligencia de un input constante de información, permitiéndole mejorar sus algoritmos en aspecto de entendimiento y reconocimiento de imagen. Por otra parte, la introducción de la inteligencia artificial en los sistemas de visión permite dotar a un sistema de un control mucho más preciso, permitiendo detectar y reconocer variables diferentes entre sí, como puede ser el reconocimiento de rostros humanos en imágenes.

Uno de los principales proyectos de la robótica y la programación es poder asemejar a los robots lo máximo posible a los seres humanos, y uno de los campos más importantes para lograrlo es la visión artificial, permitiendo dotar a los robots de ese sentido tan importante para los humanos.

1.3. ESTADO DEL ARTE

Debido a la importancia que está tomando hoy en día este campo, podemos encontrar una gran variedad de trabajos de carácter académico, no académico y profesional sobre la robótica, la visión artificial y,sobre todo, la implementación simultánea de ambos campos.

1.3.1. Proyectos profesionales

Existe una gran diversidad de información sobre este tema, tanto en forma de artículos como de libros. Entre los más interesantes a mencionar.

Encontramos el proyecto titulado “ Machine Vision in the Context of Robotics: A Systematic Literature Review ”[20], el cual se trata de un texto publicado en 2019 que se centra en analizar gran parte de la evolución de la visión artificial en los últimos 10 años, para poder poner en evidencia las mejora producidas así como los objetivos pendientes.

El paper titulado “Artificial Vision in Road Vehicles”[21]. Un trabajo que analiza el uso de la visión artificial para cumplir las funciones de sensorización necesarias que debería tener un vehículo autónomo. Como son, seguir líneas, detección de obstáculos o descripción y clasificación de objetos.

El artículo titulado “ Robot vision system based on information visualization algorithm”. Un artículo que presenta un conjunto de métodos secuenciales de visión artificial con el objetivo de conseguir un sistema más adaptativo en cuanto al control por visión artificial.

Estos tres documentos son una pequeña pero valiosa muestra de la importancia y relevancia de esta disciplina en la actualidad del mundo científico y presenta no sólo una imagen actual del control mediante visión artificial, sino su relevancia como una herramienta muy versátil en muchísimos ámbitos del desarrollo.

1.3.2. Trabajos académicos

La importancia y la versatilidad de este tipo de sistemas permite encontrar una gran diversidad de proyectos a nivel académico, no solo grandes proyectos a nivel profesional. Alguno de los más interesantes que se pueden mencionar son:

- El trabajo de final de grado de Doménech Jara, Jorge [7]. El cual se centra en el diseño de un robot de seis grados de libertad y su posterior utilización para una aplicación de pick & place. De este trabajo destaca el uso de la cámara ESP 32-CAM, mediante el IDE de arduino, pero el control del robot y el procesamiento de la imagen se hace mediante el software Mathlabs.

- El trabajo de final de grado de Berjón Valles, Ana [8]. Presentado en el año 2019 en ICAI – Universidad Pontificia Comillas. El cual se centra en la implementación de algoritmos de visión artificial para un robot ABB real, de seis grados de libertad, para la identificación y separación de piezas por colores y formas. Utilizando el software de Mathlabs.

- El trabajo de final de grado de Baixas Durbán, Rocío[9]. Presentado en el año 2020 en ICAI – Universidad Pontificia Comillas. Este trabajo implementa un sistema de visión artificial y brazo robótico, pero el control del brazo lo realiza el usuario, no lo realiza de forma autónoma el robot.

Como se puede apreciar, ya se han realizado una gran variedad de trabajos, proyectos y estudios partiendo de la idea de la unión de la visión artificial con la robótica. Sin embargo, pese a que el planteamiento es similar, los elementos empleados, y el desarrollo de los distintos trabajos permiten llegar a conclusiones muy diferentes y obtener proyectos tan variados como variadas son sus múltiples aplicaciones. De ahí, la originalidad y distinción que tiene este proyecto sobre el resto, buscando, no solo el desarrollo de un sistema funcional, sino hacerlo de manera económica y que posteriormente pueda ser implementado como una tarea académica.

1.4. ANTECEDENTES DEL PROYECTO

Como se ha indicado en los objetivos del proyecto, este trabajo busca crear una actividad académica que unifique el contenido de dos asignaturas. Al ya existir un proyecto realizado por los alumnos en la asignatura sistemas robotizados se partirá de ese trabajo. Por lo que, a continuación, se presentarán todos los recursos a nivel físico, conceptual y de programación, que se emplearán para el desarrollo del proyecto,quedando fuera del mismo el análisis teórico y la programación de las funciones de movimiento del robot, así como el diseño y montaje del mismo.

1.4.1. Brazo robot paralelogramo

El robot utilizado será un brazo robot paralelogramo. Este tipo de robots se caracteriza por estar formados por una cadena cinemática cerrada en paralelo. ya que el grado de conectividad de algunos de los eslabones es 3 o superior. Además solo tres de las articulaciones son activas, siendo de tipo de revolución,mientras que el resto de las articulaciones son pasivas.

Este tipo de estructuras en paralelogramo garantiza que el efector final esté siempre en horizontal. Debido a esto, podemos afirmar que se trata de un brazo robot con tres grados de libertad, ya que, permite desplazamientos en los ejes X, Y y Z, pero no permite ningún tipo de orientación de la pieza.

Por lo que respecta a sus propiedades constructivas, es un robot de pequeño tamaño, 16 cm de alto y 23 cm de largo en su posición de reposo. Fabricado en contrachapado, con tornillería de plástico y aluminio.

El desplazamiento del robot se consigue mediante cuatro servos modelo MG90S, conectados mediante una extensión shield a un arduino nano,por lo que, la programación de los algoritmos de movimiento del robot está hecha en este lenguaje.

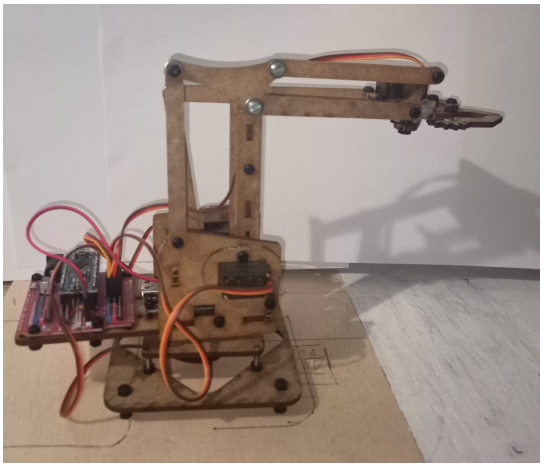


Figura 7. Imagen real del robot en posición de reposo, con el que se ha realizado el proyecto

Por lo que respecta al espacio de tarea, el robot puede alcanzar como máximo objetos a 15 cm de la base y como mínimo a 12 cm, debido al tamaño de la pinza. La altura máxima que puede alcanzar son 18,5 cm y puede desplazarse a izquierda y derecha 90 grados desde la posición de reposo, dándole un área lateral de trabajo de hasta 180 grados. Con estas características sabemos cuáles son las dimensiones máximas del área de trabajo.

1.4.2. Código de movimiento el robot

El código de programación está compuesto por la parte de definición de las funciones y los servos, el apartado de control, en el que se ejecutan las funciones y las propias funciones.

La programación se ha realizado desde el IDE de Arduino en lenguaje propio de arduino, que no deja de ser una versión muy similar a c. En este proyecto se realizarán todos los procesos de control del robot desde este entorno y con este lenguaje, ya que el shield para arduino nano permite una conexión y un control de los cuatro servos de manera muy sencilla. Pese a ello, y al estar implementando el código en c, estos algoritmos serán compatibles con cualquier microcontrolador que acepte este lenguaje, com ESP 32, solo teniendo que modificar la función de la librería servo, ya que esta es propia de arduino y puede no ser compatible al cambiar la placa de desarrollo.

De la parte de definición de las funciones y los servos cabe destacar que,, el programa define la estructura “RobotServo_t” .Esta estructura permite definir el pin al que está conectado cada servo, el offsets de los servos para compensar los pequeños errores de montaje y los valores en grados máximos que puede alcanzar cada servo sin que el robot pueda colisionar con su propia estructura. Esto se hace con el objetivo de evitar el deterioro del robot.

Además en esta etapa, también se define la estructura “RobotParams_t”, la cual almacena las variables que definen el tamaño físico real de los eslabones del robot, para poderse utilizar en las funciones de movimiento.

Por último, es importante mencionar que, para transmitirle coordenadas en milímetros al robot, se hará mediante la estructura “RobotPosition_t”, formada por la componente X, Y y Z que defina cada posición.

Del apartado de control se destaca que empieza con el desplazamiento de todos los servos a la posición de partida, ejecuta todas las funciones que se le ha indicado y posteriormente se desconectan los servos, para evitar el deterioro de los mismos .

A continuación, a la hora de clasificar las funciones de movimiento del robot, hay que distinguir entre dos tipos, aquellas que se le indica la posición objetivo al robot mediante el ángulo al que se desean mover los los servos y aquellas funciones a las que se le ha de indicar la posición deseada en coordenadas X, Y y Z.

Por lo que respecta a las funciones por ángulos encontramos, la función “writeServo”, la cual se le indica un servo y un ángulos y sitúa el servo en ese angulos y la función “moveAbsJ”, esta función permite trabajar simultáneamente sobre todos los servos, pero en necesario pasarle, tanto los ángulos de partida en los que está el robot, como los ángulos objetivo y el tiempo que se desea que tarda en realizar este desplazamiento.

Por el lado de las funciones que trabajan mediante posición, también encontramos dos, la función “moveJ” y “moveL”. Ambas funciones tienen como parámetros de entrada, la posición de partida de los ángulos en grados, el tiempo deseado para realizar el desplazamiento y la posición en coordenadas X, Y y Z objetivo. La diferencia entre ambas reside en el tipo de desplazamiento.

La función “moveJ” implementa un tipo de desplazamiento en el cual el robot no sigue necesariamente una línea recta. Se utiliza para movimientos completos del robot ya que es más rápido, pero no es recomendable para la aproximación a objeto ya que es algo menos preciso al ser movimientos no lineales.

La función “moveL” implementa un tipo de desplazamiento en el cual el robot sigue necesariamente una línea recta. Este tipo de movimiento es más recomendable para la aproximación a los objetos, ya que su movimiento es más controlado, por el contrario es algo más lento que el “moveJ”.

Adicionalmente a esto, el programa cuenta con muchos otros elementos adicionales, para asegurar el correcto funcionamiento del mismo. Pero, debido a que este código está fuera del análisis del trabajo, sólo se definen estos elementos que serán los utilizados en este proyecto.

1.5. MATERIAL Y MÉTODOS

Dentro de este punto, se presentan todas las posibles opciones planteadas a la hora de seleccionar los diferentes componentes que conforman el proyecto, acompañados de una breve descripción de los mismos

Tras esto, se presentan las ventajas e inconvenientes, que tiene cada componente para su labor concreta y se indica en base a esas características el porque se ha elegido dicho componente.

1.5.1. Material de fabricación piezas de color

Para las pruebas se han de utilizar piezas pintadas para poder ser detectados por la cámara y es importante, a su vez, que el brazo robot sea capaz de moverlos. Por lo tanto, es importante elegir un material para fabricar esas piezas que pueda cumplir con estas necesidades:

- **Madera:** en forma de cubos aunque también se puede encontrar en otras formas, se ha seleccionado esta ya que es más cercana a la forma deseada.

Figura 8 . Imágen cubos de madera de tamaño y forma similar al deseado . [10]

- **Poliestireno expandido:** también en forma de cubos ya que es más cercana a la forma deseada.

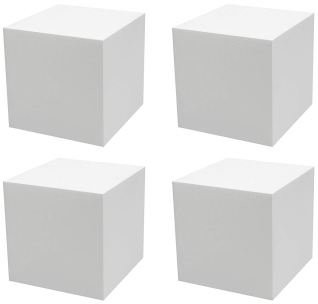


Figura 9. Imágen cubos de Poliestireno expandido de *tamaño y forma similar al deseado . [11]*

Ventajas e inconvenientes:

Madera

Ventajas	Inconvenientes
-Resistente a golpes y presión de la pinza	-Alto coste por pieza -Requiere de herramientas específicas para trabajarse -Al poseer un color propio distorsiona el color y hace muy tediosa su pintura

Poliestireno expandido

Ventajas	Inconvenientes
-Material muy económico -Fácil de trabajar y pintar -Ligero, de forma que el brazo robótico lo pueda mover sin dificultad	-Es un material que se deteriora con facilidad

Tras este análisis, la opción elegida será el **Poliestireno expandido**, ya que, su principal inconveniente se ve suplido por su bajo coste ya que en caso de deterioro es fácil y económico poder fabricar una pieza sustitutiva.

1.5.2. Cámara para la etapa de pruebas de viabilidad el proyecto

En esta parte se analizará la cámara a emplear en la primera etapa del proyecto. Donde lo importante es poder hacer una versión funcional del pick & place y lo más sencilla posible, ya que esta parte del trabajo no aparecerá implementada como tal en el proyecto final y sirve para familiarizarse con los conceptos de la visión artificial y el entorno de programación.

- **Webcam hama c-400:** se menciona este modelo ya que es del que se dispone en el momento de realizar el proyecto, pero los motivos de elección son aplicables a cualquier webcam, que solo requiere ser conectada al ordenador para que sea reconocida por el sistema operativo. No se puede procesar imágenes de manera embebida en la cámara.

Figura 10. Imágen webcam modelo hama c-400. [12]

- **ESP 32 - CAM:** se trata de una cámara de bajo coste, programable desde el ID de arduino. Puede transmitir las imágenes mediante un servidor web o por el puerto serie de arduino. Sí se puede procesar imágenes de manera embebida en la cámara.

Figura 11. Imágen cámara ESP 32 - CAM Camera. [13]

- **Raspberry Pi Camera**: esta cámara tiene un coste intermedio entre los dos modelos seleccionados, no cuenta con procesador propio y requiere conectarse a una tarjeta Raspberry para utilizarse. No se puede procesar imágenes de manera embebida en la cámara, pero si en la tarjeta Raspberry a la que se ha de conectar.

Figura 12. Imágen cámara Raspberry Pi Camera. [14]

Ventajas e inconvenientes:

Webcam hama c-400

Ventajas	Inconvenientes
-No requiere de trabajo previo antes de su implementación en el programa únicamente conectarse al ordenador -Es fácilmente implementable en python mediante las funciones de la biblioteca openCV -Se conecta al ordenador directamente con un cable USB -Buena calidad de imagen	-Tiene un precio muy elevado -Tiene un gran tamaño en comparación con las otras dos opciones

ESP 32 - CAM

Ventajas	Inconvenientes
- Muy economica -Tiene un tamaño muy reducido -Se puede alimentar directamente con el pin de 5V de la placa de arduino	-Requiere de una programación previa para transmitir la imagen mediante un servidor web -Se ha de implementar en el código de python un sistema para concentrar y procesar la información de la cámara enviada por el servidor web

Raspberry Pi Camera

Ventajas	Inconvenientes
-Facilmente concetabler al ordenador a través de la raspberry pi -Si se dispone de la raspberry pi la cámara es la mas economica perse	-El módulo de mara y la raspberry pi juntos tiene un tamaño muy grande -Requiere un paso intermedio entre la mara y el ordenador como es la raspberry -La suma de tarjeta y cámara tiene un precio muy elevado

Debido a este análisis, para esta primera etapa cuyo objetivo es hacer una versión funcional, la opción más óptima es la **Webcam hama c-400**, ya que, es la más sencilla de conectar, evitando así dificultades a la hora de realizar esta primera etapa que vengan del modelo de cámara y no del correcto funcionamiento del sistema de visión artificial.

1.5.3. Cámara para la implementación final del proyecto

A continuación, es necesario reformular las necesidades con respecto a la cámara elegida. Ya se ha comprobado la viabilidad del proyecto, por lo que, partiendo de los tres mismos modelos de cámara se volverá a plantear en análisis de ventajas e inconvenientes pero esta vez los puntos a valorar más importantes son, el coste de la cámara y sobre todo el hecho de tener un procesador integrado para poder procesar la imagen en la propia cámara y poder darle esa naturaleza de sistema embebido que busca el proyecto.

Webcam hama c-400

Ventajas	Inconvenientes
-Buena calidad de imagen	-Tiene un precio muy elevado -No se puede procesar imagen de manera embebida en la cámara

ESP 32 - CAM

Ventajas	Inconvenientes
- Muy economica -Tiene un tamaño muy reducido -Se puede alimentar directamente con el pin de 5V de la placa de arduino -La cámara cuenta con procesador integrado.	-Si se quiere hacer un procesado muy exhaustivo de la imagen o trabajar con imágenes de alta resolución su procesador puede no ser suficiente -Calidad de imagen reducida

Raspberry Pi Camera

Ventajas	Inconvenientes
-Es muy sencillo de realizar el sistema embebido ya que la raspberry pi en sí es un ordenador de pequeñas dimensiones, con potencia suficiente para un procesamiento de imagen sencillo	-El módulo de mara y la raspberry pi juntos tiene un gran tamaño -La suma de tarjeta y cámara tiene un precio muy elevado -Requiere utilizar linux como sistema operativo, lo cual a la hora de ser replicado por los alumnos puede presentar dificultades

Por lo tanto, las versiones finales del proyecto se realizarán con la **ESP 32 - CAM**, sobre todo por su bajo coste, pero también por el hecho de disponer de un procesador integrado sin necesidad de otros componentes adicionales como sería el ordenador o la raspberry pi.

1.5.4. Material para la base de fijación

Es necesario plantearse la necesidad de construir una base sobre la que fijar los distintos elementos físicos que formarán el proyecto. Tal como se indicó en las condiciones del encargo. Por eso en este apartado se analizarán los posibles materiales de fabricación para la base.

- **Aluminio**: En laminas de 4 mm de espesor y 100 mm de ancho por 100 mm de largo.

Figura 13. Imágen plancha de aluminio de 4 mm de espesor . [15]

- **Contrachapado**: también en láminas de 4 mm de espesor pero de 300 mm de ancho por 400 mm de largo

Figura 14. Imágen plancha de contrachapado de 4 mm de espesor . [16]

Ventajas e inconvenientes:

Aluminio

Ventajas	Inconvenientes
-Muy resistente -Sensación de calidad	-Alto coste por plancha -Requiere de herramientas específicas para trabajarse

Contrachapado

Ventajas	Inconvenientes
-Material económico -Fácil de trabajar y pintar	-Es un material que se deteriora con facilidad -Difícil de eliminar las marcas, salvo pintando

Basándose en la naturaleza del proyecto, el material utilizado para la base de sujección será el **contrachapado**, debido principalmente, a su bajo coste y a su facilidad para trabajar y agujerearse frente al aluminio.

1.5.5. Lenguaje de programación

Por último, se ha de seleccionar un lenguaje de programación. Se cuenta con el antecedente de que el control del robot se hace desde una placa de Arduino, por lo que no solo es necesario encontrar un lenguaje de programación que tenga soporte para análisis y tratamiento de imagen en tiempo real sino que es necesario que el lenguaje permita de manera sencilla la comunicación entre el ordenador y la placa Arduino.

- **Python:**

  Se trata de un lenguaje muy versátil , fácil de entender visualmente y fácil de depurar. Cuenta con la biblioteca “Opencv”, una biblioteca que facilita en gran medida todos las etapas que llevan consigo la visión artificial, mediante funciones sencillas, lo cual no solo hace fácil trabajar con ella sino aprender si no se tiene ningún conocimiento al respecto. A su vez, la biblioteca “Serial” permite de manera sencilla la comunicación a través del puerto serie, que puede ser en una primera instancia la forma más efectiva de comunicación entre el ordenador y la placa arduino.

  Además, debido a la relevancia de este lenguaje dentro del campo de la visión artificial, es muy sencillo encontrar cientos de proyectos e información lo cual facilita el aprendizaje del lenguaje y de su uso para tareas de visión artificial.

- **C++:**

  A nivel de desarrollo, C + + tiene las mismas ventajas que python, ya que, también cuenta con la biblioteca “Opencv” y su biblioteca propia para la comunicación serie con Arduino. Por tanto, la única diferencia si no se tiene ningún conocimiento de los mismos reside en la información que de la que se dispone para trabajar con uno frente al otro.

  En este punto, la información y los ejemplos disponibles son mucho menores, por lo que es más difícil el aprendizaje.

Ante esta situación y sabiendo que respecto a potencial para realizar el proyecto, ambos lenguajes son perfectamente viables, el único y principal motivo para decantarse por **python** en la grandísima cantidad de información de la que dispone, haciéndolo la mejor opción en la etapa de aprendizaje de este lenguaje y de su utilización en el campo de la visión artificial.

1.6. DESARROLLO DE LA SOLUCIÓN

El desarrollo completo del proyecto como se mencionó en anteriormente está dividido en 3 grandes etapas. A continuación se desglosa cada uno de los componentes de las tres etapas. (figura 15)

Figura 15. Esquema de etapas. muestra los componentes que forman cada una de ellos así como la relación entre los mismos

A continuación, y tras haberse realizado se irán presentando las soluciones desarrolladas para cumplir con los objetivos planteados para cada etapa.

1.6.1. Control no embebido mediante webcam

Antes de comenzar a programar sobre la cámara o el robot es necesario determinar la posición de los elementos. Esta se mantendrá constante durante todas las etapas del proyecto. De ahí la importancia de implementar todos los elementos necesarios de manera que no se entorpezcan unos con otros.

1.6.1.1. Posición de los elementos y fijación de la cámara

Los elementos a situar serán, el robot, la cámara y por tanto el soporte que la mantenga para tomar la imagen cenital y dos recipientes para almacenar las piezas.

El primer paso consistirá en delimitar el espacio de tarea, cercano a la base, es decir, todos los puntos a los que, por construcción, el robot puede llegar con la pinza y por tanto todos aquellos puntos en los que podría recoger una pieza. Esto se realiza de manera experimental sobre una plancha de contrachapado y marcando toda el área que el robot puede alcanzar (figura 16).

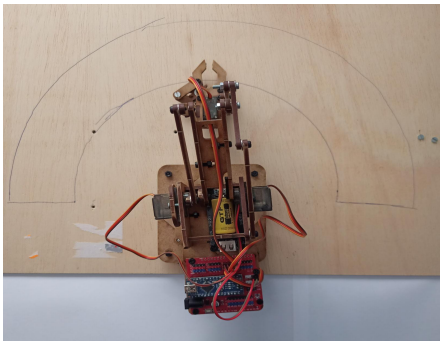


Figura 16. Imágen tomada de la prueba realizada para determinar el espacio de tarea.

Tras esto, se define el espacio de tarea como el área comprendida entre una semicircunferencia de 240 mm de radio y otra de 370 mm de radio. También se puede afirmar que el área donde el robot posee una mayor libertades es, de frente suya, por lo tanto es el mejor lugar para situar la cámara.

A continuación, se tomará el ángulo de visión de la cámara, para delimitar el área que la cámara puede detectar, y definiremos el espacio de trabajo como el área a la que puede desplazarse el robot y simultáneamente ser observado por la cámara.

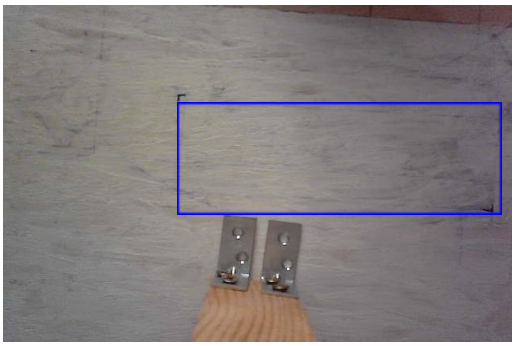


Figura 17. Imágen tomada desde la webcam para determinar la posición idónea de la misma.

Una vez realizado esto, se puede determinar que la posición idónea para la cámara es, a 150 mm del robot y a una altura de 300 mm. De esta manera obtenemos un área de trabajo rectangular de 50 x 150 mm. Como se puede apreciar en la (figura 17)

Para poder fijar la cámara se construirá un soporte mediante un listón de madera de 300 x 40 mm, cuatro escuadras en L de 25 mm y 8 tornillos de acero galvanizado para madera, siguiendo las dimensiones definidas en el plano 4 del apartado de planos del proyecto.

Tras esto, se fijarán el soporte para la cámara y el robot a la base, utilizando tornillos y tuercas de M3. Empleando los agujeros definidos en el plano 3 del apartado de planos del proyecto.

Adicionalmente, se situaran los dos recipinetes para piezas a la derecha del robot a 80 mm, para evitar que el color interfiera con la detección de color de la camara. Estos recipientes tendrán el tamaño y la forma definida en el plano 2 del apartado de planos del proyecto.

1.6.1.2. Programación en python para la detección de piezas

El código en python se puede dividir en dos etapas, una primera etapa basada en la adquisición de datos, la detección de las piezas y el tratamiento de la imagen y una segunda basada en el análisis y cálculo de esos datos obtenidos para posterior envío a través del puerto serie. Este funcionamiento se puede expresar de manera esquemática (figura 18).

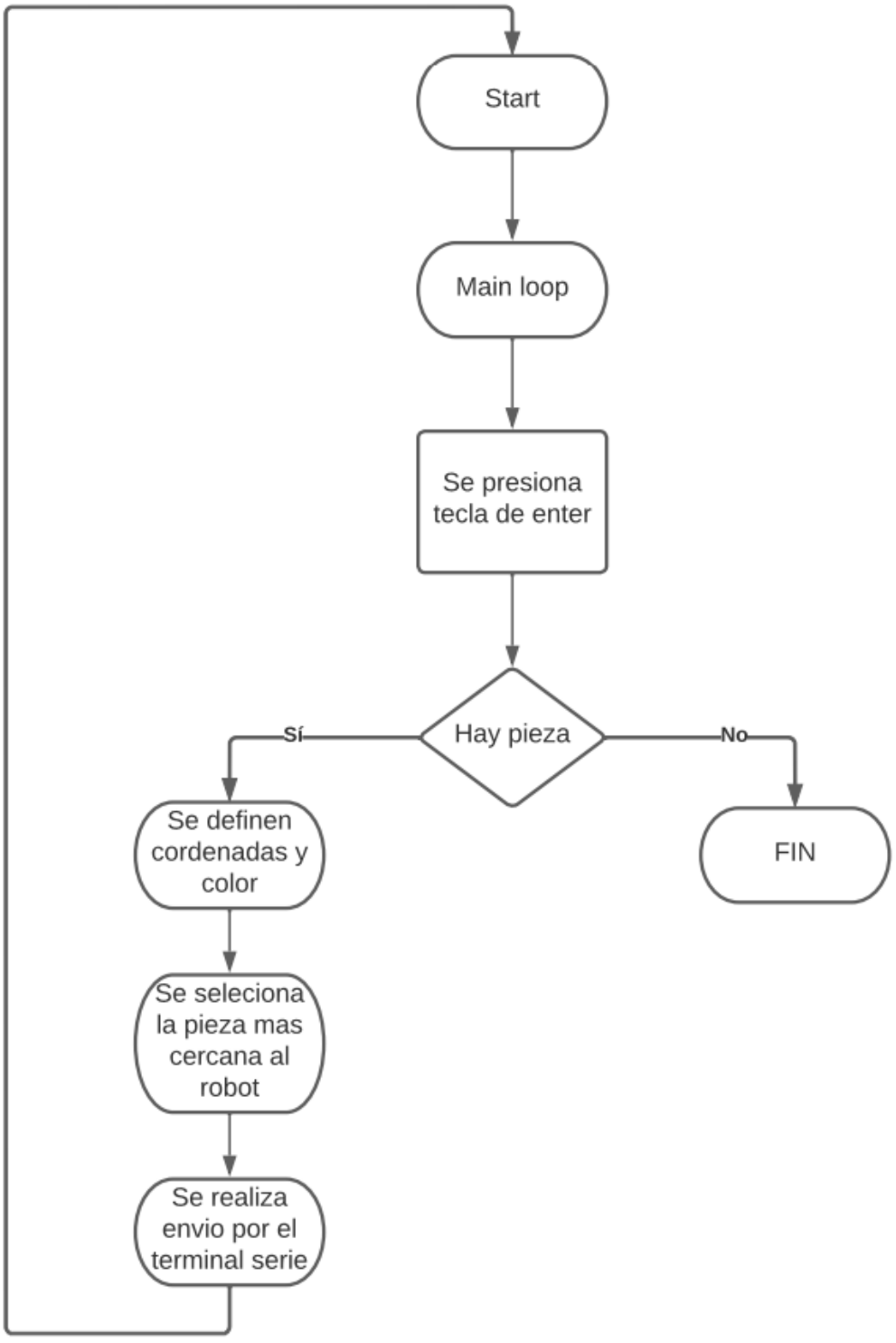


Figura 18. Diagrama de funcionamiento del programa de python para la detección de piezas

El primer paso será, mediante la librería serial, establecer la comunicación serie con arduino. En esta implementación, python solo será emisor y arduino receptor, para facilitar el funcionamiento. El arduino está conectado en el puerto seis, por lo que definiremos este puerto para realizar la conexión. Como se puede apreciar en el siguiente fragmento de código (figura 19).

```
#------------------------Conexión Arduino---------------------------------
serialArduino = serial.Serial("COM6", 9600)
```

Figura 19. Fragmento del código utilizado para definir el puerto y la frecuencia de conexión de python con arduino a través del puerto serie . Extraído del código del anexo A-2

Para el análisis de imagen utilizaremos la librería Opencv. Para la deteción de color Opencv utiliza el modelo de colores HSV, y no el RGB propio de las imagenes digitales. Esto es debido a que este modelo permite definir colores como rangos dentro de un espectro y no como combinaciones de las componentes rojo, verde, y azul, como pasa con el RGB. Lo cual lo hace más fácil de interpretar. Por lo que el primer paso será identificar los rangos a los que pertenecen los colores a tratar en este modelo, esto se puede hacer mediante la interpretación de una tabla (figura 20) que muestra el espectro completo de colores.

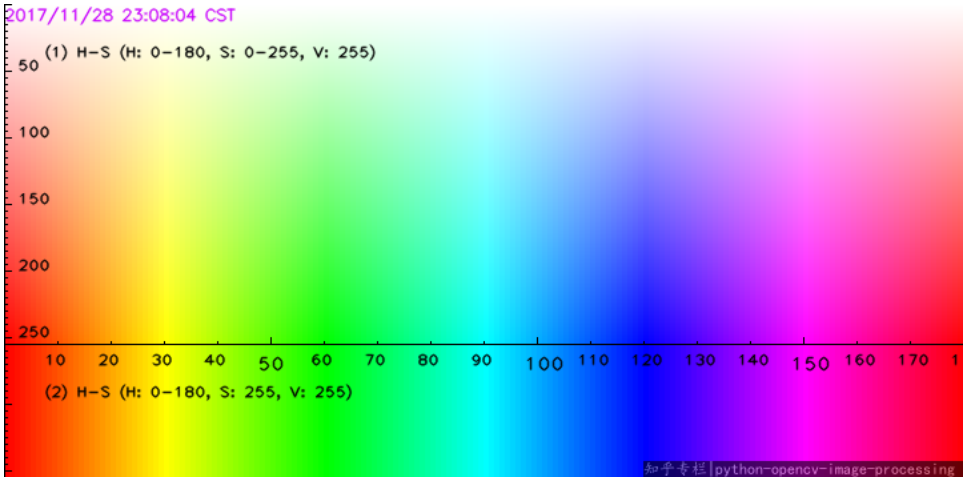


Figura 20. Imagen de espectro de colores para HSV . [17]

La componente en X de este diagrama permite definir el valor de "*Hue" (matiz)*, que es el color propiamente dicho. La componente Y por el contrario permite definir los valores de "Saturation" y "Brightness" (Saturación y brillo), es este proyecto se cogerá todo el rango de brillos comprendido del 20 al 255 y de saturación del 100 al 255. Ya que los colores fuera de estos rangos son tonos demasiado claros para el tipo de detección que se busca. En muchos casos acercándose al blanco, lo cual podría interferir con el cuerpo central de las piezas que ha de detectar el programa.

Tras esto se definirán los límites de estos rangos en el programa. En el caso del color rojo tiene dos segmentos en el espectro, así que se definirán ambos y posteriormente se sumarán, definiendo el rojo como el color resultante de la suma de ambos segmentos. Como se puede apreciar en el siguiente fragmento de código (figura 21).

```
#-------------------------Definimos los rangos de detección----------------
azulBajo = np.array([100, 100, 20], np.uint8)
azulAlto = np.array([125, 255, 255], np.uint8)

redBajo1 = np.array([0, 100, 20], np.uint8)
redAlto1 = np.array([10, 255, 255], np.uint8)

redBajo2 =np.array([170, 100, 20], np.uint8)
redAlto2=np.array([180, 255, 255], np.uint8)
```

Figura 21. Fragmento del código utilizado para definir los límites altos y bajos ,en HSV, de los colores a detectar. Extraído del código del anexo A-2

El siguiente paso será convertir la imagen tomada a HSV. Es interesante indicar que Opencv a la hora de tomar las imágenes de la webcam, no lo hace en el formato tradición RGB sino que lo hace en BGR, esto es importante ya que a la hora de realizar la conversión, mediante la función "cv2.cvtColor" introduciremos como parámetro "cv2.COLOR_BGR2HSV", ya que de lo contrario, los colores de la imagen se verán distorsionados.

A continuación, se crearán variables con la detección de los colores según los límites definidos anteriormente mediante la función "cv2.inRange" y sobre esas variables detectadas se definirá el contorno por colores mediante la función "cv2.findContours". Como se puede apreciar en el siguiente fragmento de código (figura 22)

```
# ----------------------Deteción de color en la imagen--------------------
    maskAzul = cv2.inRange(HSV, azulBajo, azulAlto)

    maskRed1 = cv2.inRange(HSV, redBajo1, redAlto1)
    maskRed2 = cv2.inRange(HSV, redBajo2, redAlto2)
    maskRed = cv2.add(maskRed1, maskRed2)
#----------------- Detectar el contorno del color-----------------------
contornoazul,_ = cv2.findContours(maskAzul, cv2.RETR_EXTERNAL, cv2.CHAIN_APPROX_SIMPLE)

contornorojo,_ = cv2.findContours(maskRed, cv2.RETR_EXTERNAL, cv2.CHAIN_APPROX_SIMPLE)
```

Figura 22. Fragmento del código utilizado para definir las mascaras de color segun los limites definidos y posteriormente detección de los contornos en función de esos colores. Extraído del código del anexo A-2

Sobre estos contornos se obtendrá su centro de gravedad mediante "cv2.moments", así como las coordenadas x e y del mismo. Tras esto se dibujara el centro en la imagen y se mostrarán por pantalla las coordenadas y el color (figura 23). Esto se hará para el contorno rojo y el azul de forma separada.

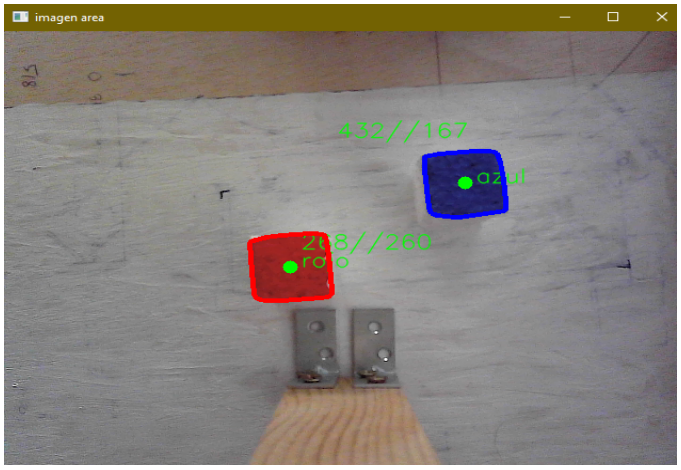


Figura 23. Imagen tomada desde la webcam que ilustra la detección de colores y la obtención de las coordenadas x e y de los centros de gravedad de las piezas

Conociendo los valores de las coordenadas x e y sobre el espacio de trabajo, para cada color, se fijará como prioridad aquella pieza con coordenada y menor. Tras esto, se pasará de píxeles a centímetros mediante la ecuación de la recta. Siendo los centímetros a desplazar el valor que se le pasará al robot, los centímetros totales el máximo que se puede desplazar el robot en ese eje y los valores de x e y máximos y mínimos los límites de la zona de trabajo del robot.

$$cm\ a\ desplazar\ en\ X\ =\ \frac{cm\ totales\ en\ X}{Xmax - Xmin} * (posición\ pixeles\ en\ X\ -\ Xmin)$$

Aplicaremos esta ecuación tanto para lo coordenada en x como en y.

Por último, para la transmisión por el puerto serie se dividirán las cifras en unidades, decenas y centenas, ya que se enviaran los datos byte a byte, todos codificados en lenguaje ascii. Cada cadena empezará con una "@" seguido de los dos valores x e y separados en unidades, decenas y centenas. A continuación, un byte de número que proporciona la información sobre el color de la pieza, el 0 para el azul y el 1 para el rojo y por último una "*", que indica que el envío de información ha sido completado y se puede comenzar con el desplazamiento del robot. Esto se puede apreciar de manera más sencilla a partir del siguiente diagrama (figura 24).

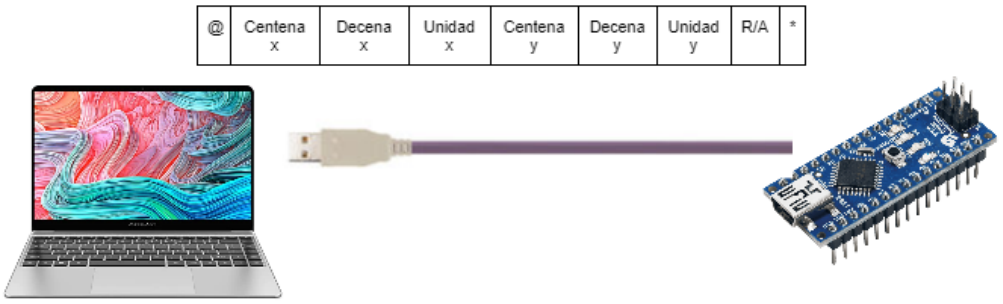


Figura 24. Esquema ejemplo cadena de información enviada por el ordenador al arduino para el control del robot

Cabe destacar, que como se indica en el apartado de condiciones del proyecto, el robot ha de ser capaz de distinguir entre dos piezas de colores distintos simultáneamente. El criterio implementado para determinar en esa situación cuál ordenar primero es aquella más cercana al robot, es decir, una componente Y menor, ya que de esta manera es más difícil una posible colisión del robot con la pieza más alejada.

Adicionalmente, el programa cuenta con una función de calibrado, que modifica los valores de x e y máximos y mínimos, para poder compensar los pequeños desplazamientos que pueda sufrir la cámara.

Para el calibrado se empleará la pieza azul, situándose en la esquina inferior derecha y presionando la tecla “y”, a continuación en la esquina superior izquierda, y presionando la tecla “v”.

Cabe destacar que el modo de calibración cuenta con un sistema de seguridad, de manera que, si se realiza un mal calibrado da un error por pantalla y vuelve a la calibración por defecto, para evitar errores que puedan producir deterioros en el robot.

1.6.1.3. Programación en Arduino para la interpretación de datos y el desplazamiento del robot

El código de arduino partirá como se ha mencionado en el apartado de antecedentes del proyecto, parte de un programa que ya implementa las funciones de movimiento del mismo. Partiendo de esto es necesario desarrollar dos elementos más, por una parte la parte del programa que hará de receptor y por otra la implementación de las funciones ya definidas para lograr la tarea de almacenaje deseada.

Para la parte del programa que hará de receptor partiremos de la idea de conocer cómo es la estructura de envío de datos. Por lo que en la parte del bucle del código, mientras la comunicación serie esté habilitada. Se leerá byte por byte el buffer del puerto serie, de forma que siempre que se respete la estructura de envío y no hayan interferencias se obtendrá al final una estructura con las componentes X e Y donde se encuentra el objeto. (figura 25).

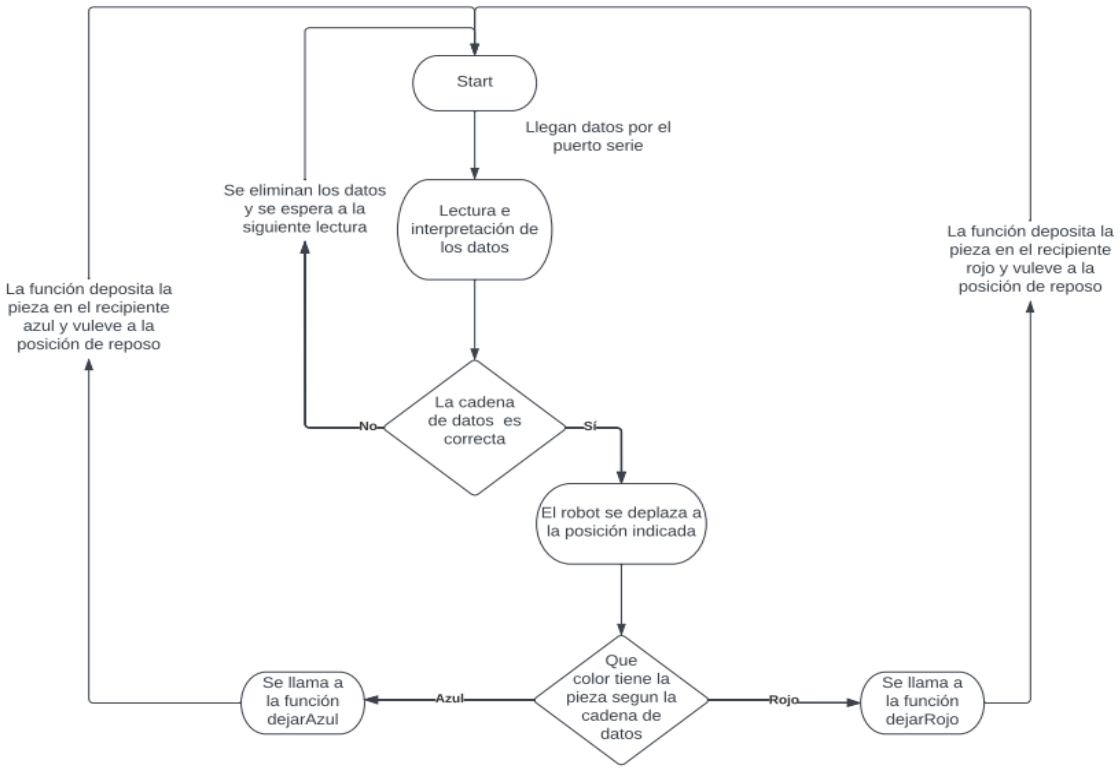


Figura 25. Diagrama de funcionamiento del programa de Arduino para el control del movimiento del robot

Como también se indicó en el apartado de estado antecedentes del proyecto, las estructuras de desplazamiento que se usan como variables para las funciones, no sólo tienen componente X e Y, sino también componente Z. En este caso, y sabiendo que la forma y el tamaño de las piezas es siempre el mismo, la componente Z será constante y estará fijada a 30.36 mm, lo que corresponde aproximadamente con algo más de la altura media de la pieza, ya que esto facilitará su agarre.

En este punto del desarrollo del código, ya no solo se conoce la posición de la pieza, sino que también se conoce el color de la misma, ya que es uno de los elementos que se ha enviado a través del puerto serie. Por lo que sabiendo esto y el hecho que la posición de los recipientes es fija se implementarán dos funciones, “dejar rojo” y “dejar azul”,cuyo funcionamiento consiste en , desplazar el robot desde donde hasta encima del recipiente del color en cuestión, abra la pinza dejando caer la pieza y vuelva a la posición de reposo, de manera que permanezca preparado por si se envía a través del puerto serie la posición de otra pieza. Esto se puede apreciar de manera más sencilla a partir del siguiente diagrama (figura 21).

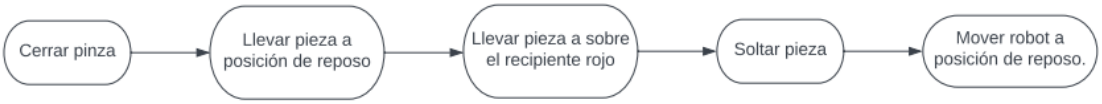


Figura 26. Diagrama que muestra los movimientos realizados en la función dejar rojo.

Al llegar al último byte del buffer, según la forma de envío, debería corresponder con el símbolo del asterisco. Si esto se cumple, el robot abrirá la pinza y empezará su desplazamiento hasta la posición designada. Una vez llegado al punto designado comprobará el color que se le ha indicado por el puerto serie para la pieza y llamará a la función “dejar” que corresponda según el color. Tras esto devolverá los valores X e Y a su valor inicial, para evitar interferencias con posibles iteraciones del bucle.

Tras la implementación de todo, se consigue un sistema funcional que cumple con las características deseadas para la primera etapa del proyecto. En la siguiente imagen se puede apreciar una imagen del proyecto ya implementado funcionando(figura 27).

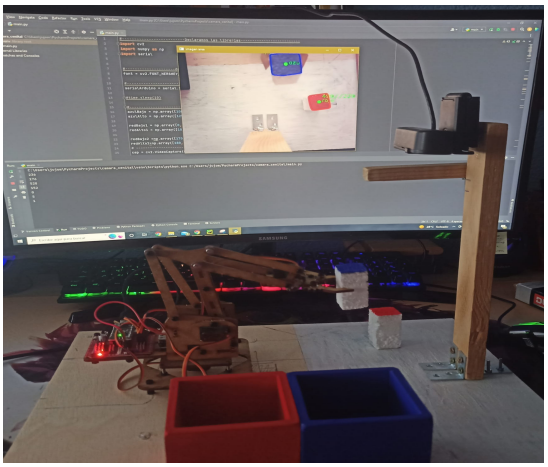


Figura 27. Foto del sistema resultante de la etapa uno funcionando tras la integración de todos los elementos.

1.6.2. Control no embebido mediante ESP 32-CAM

En esta etapa, el primer paso será determinar el papel que va a cumplir la cámara ESP 32 y en qué va a diferir frente a utilizar la Webcam como se ha hecho en los apartados anteriores. Por otra parte, la función de la cámara serán dos, por un saludo, tomar las imágenes, sin ningún tipo de tratamiento y por otro será enviar estas imágenes en tiempo real a través de un servidor web. Esto se ejemplifica mediante el siguiente diagrama de bloques (figura 28).

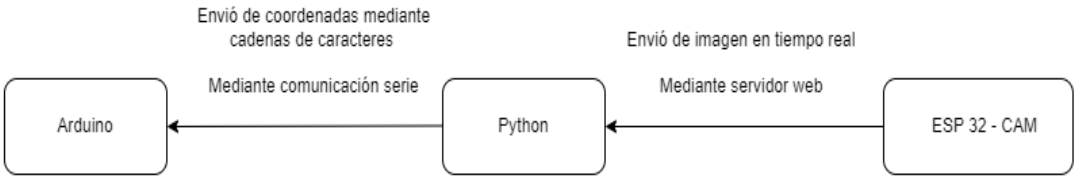


Figura 28. Diagrama de comunicación entre los componentes del sistema

Por lo que, a grandes rasgos, en este punto se desarrollará como programar la ESP32-CAM, para que tome las fotos y las envíe y por otro lado, modificar el programa de python para que pueda recibir esas imágenes e interpretarlas.

Antes de poder utilizar el módulo ESP 32 -CAM como servidor web, es necesario programar el módulo para conseguir este propósito, esto se hará mediante el proceso descrito en el Anexo A-7.

Una vez que se dispone de la cámara funcionando como servidor web, se analizaron los cambios que se han de realizar en el código de python, ya que al contrario que sucedía usando la webcam, en esta etapa el proyecto el programa a de ser capaz de conectarse a la dirección web en la que emite la cámara y obtener de esa dirección la imagen en tiempo real a procesar.

El primer paso consistirá en acceder a la dirección de la cámara desde el programa de python. Para eso se deberá importar la librería “urlopen”, que es parte del paquete “urllib.request”.

A continuación, se definirá la variable stream, como la url de conexión de la cámara, que proporciona el monitor serial de arduino al conectarlo a la red wifi. Adicionalmente a esta url se le ha de añadirse “:81”, que como indica el monitor serie es el puerto habilitado para el stream de la imagen y “/stream” para leer solo la imagen y eliminar el interfaz de control de la imagen. Ya que el tratamiento de la imagen se hará desde python.

También se ha de definir una variable de tipo bytes(), donde se almacenará la información leída de la url.

El siguiente paso consistirá en leer de la url mediante la función “stream.read” y buscar dos bytes concretos que se utilizarán más adelante para definir la dimensión de la imagen

Con estos bytes conocidos y la lectura completa de la url se definirá la variable “ frame” que se trata de la imagen en tiempo real que está tomando la cámara.Como se puede apreciar en el siguiente fragmento de código (figura 29). A partir de esta variable se trabajará y se realizarán el resto de procesos.

```
#-------------------------lectura de la imagen-------------------------
while True:
    bytes += stream.read(1024)
    a = bytes.find(b'\xff\xd8')
    b = bytes.find(b'\xff\xd9')
    if a != -1 and b != -1:
        jpg = bytes[a:b + 2]
        bytes = bytes[b + 2:]
        if jpg:
                                frame = cv2.imdecode(np.fromstring(jpg, dtype=np.uint8),
cv2.IMREAD_COLOR)
```

Figura 29. Fragmento del código utilizado para leer de la url y buscar los bytes de interés. Extraído del código del anexo A-4

Estos dos fragmentos de código sustituyen a las funciones de “video captur” y “read” que se han empleado en el apartado anterior del proyecto.

Por lo que, con este cambio el programa ha pasado de tomar la imagen a través de la conexión USB de la webcam a hacerlo leyendo del servidor web generado con la ESP 32-CAM.

Aun así, es necesario realizar una serie de cambios adicionales en el código, esto son pequeños ajustes en valores que interactúan directamente con la imagen, ya que al cambiar de cámara, algunas propiedades como tamaño el tamaño de la imagen y el ángulo de visión se han visto afectados, pudiendo interferir negativamente con el correcto funcionamiento del programa.

El primer cambio corresponde con el valor de área mínima que se define para considerarlo una pieza y en consecuencia trazar su perímetro y calcular su centro de masas. En el código anterior este valor era de 3000 unidades de área y tras implementar la ESP 32 - CAM se deberá fijar en 1000 ya que de lo contrario no detectará ninguna pieza. Esto se ha de hacer igualmente para el área roja y el área azul. En la siguiente imagen se muestra la detección del contorno de una pieza tras las modificaciones pertinentes en el código (figura 30)

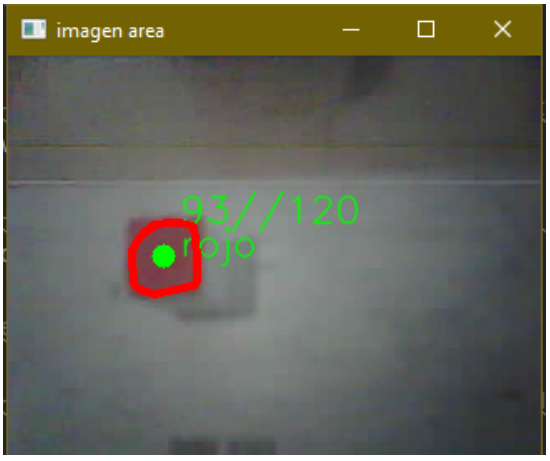


Figura 30. Imagen tomada desde la ESp 32-CAM que ilustra la detección de colores y la obtención de las coordenadas x e y de los centros de gravedad de las piezas

Al mismo tiempo, debido a esta representación más pequeña que hace la cámara de los objetos puede ser necesario modificar los valores de x e y máximos y mínimos que se utilizan para calcular el desplazamiento. Para este caso particular se emplea un rango de X de 118 a 137 y de Y de 29 a 296. Esto se puede hacer ,a su vez, empleando la función de calibrado con la que cuenta el código.

Con estas modificaciones se consigue replicar la función de pick & place mediante detección de color que se habia conseguido en el apartado anterior pero emplenado una camara de bajo coste, consiguiendo así, uno de los principales objetivos, como era realizar esta tarea de manera económica y fácilmente replicable. A continuación, se muestra una imagen del proyecto ya implementado funcionando (figura 31).

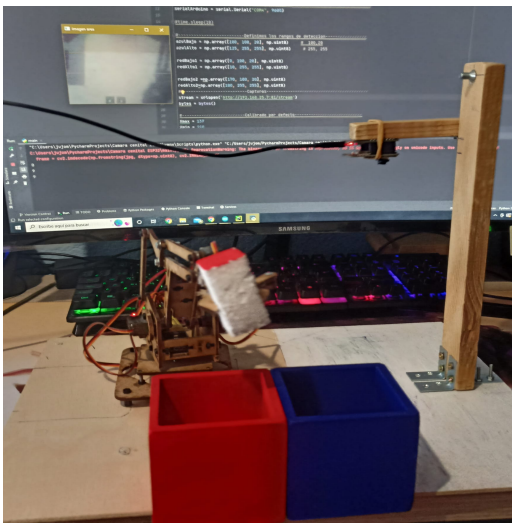


Figura 31. Foto del sistema resultante de la etapa dos funcionando tras la integración y adaptación de todos los elementos.

1.6.3. Control embebido mediante ESP 32-CAM

Esta última etapa se trata del punto más experimental, se busca programar, mediante algoritmos en bajo nivel en c, el comportamiento obtenido en las versiones anteriores, pero esta vez sin contar con el soporte de bibliotecas externas para el tratamiento de la imagen, sino desarrollando los algoritmos de la manera más transparente posible.

Es importante mencionar que existen distintas bibliotecas que implementan funciones, las cuales facilitan algunas etapas del procesado de la imagen, para este punto no se usarán, ya que lo que se busca es familiarizarse con la lógica y el funcionamiento que hay detrás de esas funciones y ser capaz de desarrollarlas de la manera más independiente posible.

Además, existe otro motivo relacionado con la capacidad de almacenaje de la cámara, el incluir librerías enteras podría llegar a sobrepasar el almacenaje completo de la cámara, lo cual es el otro motivo por el que se descarta el uso de librerías.

Pese a ello, si será necesario utilizar la librería “Eloquent visión” [24], en particular sus apartados “IO” y “ImageProcesing”. Estos se usarán para la configuración de la cámara y para la lectura y posterior escritura del buffer de la cámara que almacenará la imagen tomada.

Una vez definida la forma de trabajo en este punto, se procederá a definir las etapas en las que se van a dividir el procesado de la imagen (Figura 32), para después analizarla etapa a etapa.

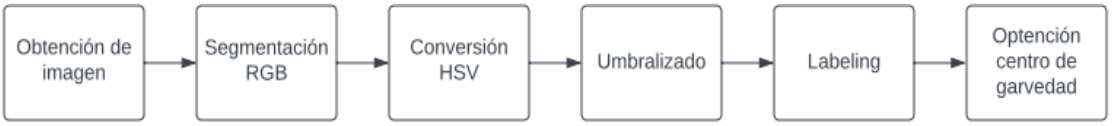


Figura 32. Diagrama de las etapas de procesador de imagen

1.6.3.1. Programación de la etapa de obtención de imagen

Como se ha presentado, en la introducción esta etapa se hará apoyándose en la biblioteca “Eloquent vision”[24]. Para eso, antes es necesario definir el modelo de cámara, en este caso “CAMERA_MODEL_AI_THINKER”, Así con el tamaño de la imagen, para la cual se usará “ FRAMESIZE_QVGA”, el cual cuenta con una anchura de la imagen de 320 píxeles y una altura de 240. Los valores de altura y anchura también se definirán como constantes, porque aunque no se utilicen en en esta etapa será un elemento recurrente en el resto de etapas.

A continuación, se iniciará la cámara mediante la intrusión “camara.begin” pasándole como parámetro primero el tamaño de la imagen y luego el formato de los píxeles. Para este caso se usará el formato ”PIXFORMAT_RGB565”, cada píxel estará definido en formato RGB, siendo los 5 primeros bytes para definir el rojo, los siguientes 6 para el verde y los últimos 5 para el azul. Todo esto en la etapa de “setup” del código.

Por último, una vez iniciada la imagen y ya en el loop principal del código, se obtendrá la imagen mediante la función “camara.capture()” y se almacenará en un vector de punteros del tipo “camara_fb_t” llamado “frame” .

Es importante mencionar que para más adelante poder extraer la información en el vector “frame” se tendrá que hacer mediante la función “buff[]”, indicando la posición del byte que se desea leer.

1.6.3.2. Programación de la etapa de segmentación RGB

Antes de proceder con la segmentación propiamente dicha hay que tener en cuenta dos aspectos cruciales para entender el funcionamiento de la imagen. Como se ha definido, en el apartado anterior, el formato de los píxeles será “rgb565”, eso implica que para cada pixel se necesitan 16 bits, esto es relevante ya que la imagen guardar en en el puntero “frame” está dividida en bytes, por lo que cada píxel se define como la suma de los dos bytes consecutivos del vector “frame”.

Para recorrer este vector, se implementará un bucle for, que vaya desde la posición cero hasta la posición correspondiente al doble del tamaño de la imagen. Esto es debido a lo mencionado anteriormente de que cada píxel se corresponde con dos bytes. Dentro ya del bucle for, se leerán los dos bytes que toquen, se sumarán aplicando aritmética de punteros, es decir primero desplazando el superior ocho posiciones y luego sumando mediante la operación “or”.

A continuación, ya contando con ese píxel completo, se procederá a la segmentación propiamente dicha, aplicando de nuevo aritmética de punteros, se dividirá cada píxel en sus tres componentes R, G y B. Como se puede apreciar en el siguiente fragmento de código (figura 33).

```
uint8_t r = (pixel & 0b1111100000000000) >> 11;
uint8_t g = (pixel & 0b0000011111100000) >> 5;
uint8_t b = (pixel & 0b0000000000011111);
```

Figura 33. Fragmento del código utilizado para separar por aritmética de punteros las componentes r,g y b de cada píxel que compone la imagen . Extraído del código del anexo A-5

1.6.3.3. Programación de la etapa de conversión de RGB a HSV

Una vez segmentados los bits que componen cada pixel, se buscará pasar a HSV, para continuar con la forma de trabajo que se ha seguido durante todo el proyecto. El único elemento diferenciador es que, en esta etapa solo tendremos en cuenta el tinte, es decir la componente H, de forma que, se considerara rojo a azul cualquier tinte que esté dentro del rango, independientemente de su saturación o su brillo.

Para ello, se ha tomado como referencia el proyecto de George Runielli [18], el cual presenta una función que permite convertir partiendo de las componentes de RGB a las de HSV. Este código se ha modificado y ajustado al proyecto para conseguir combertir de RGB a solo H, el funcionamiento a grandes rasgos consiste en ir determinando el peso de las tres componentes de color para determinar si se encuentra dentro de los rangos que determinan por el tinte si es un color o otro.

Cabe destacar, que la función toma como entradas las componentes RGB en un rango del 0 al 255. En la etapa anterior los rangos de las componentes son del 0 al 31 para el rojo y el azul y de 0 a 63 para el verde, por lo que, antes de poder llamar a la función será necesario una adaptación de los rangos, ya que de lo contrario la conversión no será correcta.

1.6.3.4. Programación de la etapa de umbralizado

Antes que nada, hay que definir en qué consiste el umbralizado, esta etapa se centra en crear una imagen en balnco y negro a partir de la imagen real. Siendo los pixeles negros aquellos del color a umbralizar y el resto blancos, esto se hace ,principalmente, para poder determinar mas adelante la silueta de las formas y que no interfieran el resto de pixeles que no son de interes.

El primer paso será determinar cuáles son los colores a umbralizar y sus rangos de h, ya que será este valor el que utilizaremos para determinar si un píxel es de un cierto color u otro. Aunque de manera teórica y como ya hemos utilizado en este proyecto, los rangos de H que determina el azul y el rojo son de 100 a 125 para el azul y de 0 a 10 para el rojo, mediante pruebas experimentales se determina que para esta cámara los rangos en los que trabajar pasan a ser de 0 a 50 para el rojo y de 230 a 255 para el azul. Esto es debido ,principalmente, a que el rango de valores de salida de la función de conversión es de 0 a 360.

Cabe destacar que, para guardar la imagen umbralizada se usará vector de vectores, que funcione al uso práctico como una matriz. Esto por dos motivos fundamentales, el primero es poder situar los objetos en el espacio, ya que, a la hora de determinar el centro de gravedad será necesario que la imagen está compuesta en 2D, ya que la posición se obtendrá con relación a la altura y la anchura de la imagen, cosa que no se puede hacer con una imagen como vector. La segunda está relacionada con la memoria de la ESP 32-CAM, para umbralizar dos colores se tienen que definir una matriz para cada color. Debido a las limitaciones de memoria del dispositivo, si esto se hiciera directamente sobre memoria física no cabría, de forma que es necesario utilizar memoria dinámica, empleando la función malloc para definirse.

Por lo tanto en el proceso de umbralizado, se lee cada pixel de la imagen, se determina su color, si es rojo se guarda un 255 en la matriz roja y un 0 en la azul y viceversa si es azul, por el contrario si el pixel no es de ninguno de los dos colores se guarda un 0 en ambas.

Adicionalmente, al escribir sobre las matrices umbralización por color se llevará una cuenta del número de píxeles de cada color. Esto se hará ya que se usará esa cuenta para determinar de qué color es la pieza de la imagen, ya que el número de píxeles del color de la pieza deberá ser mucho mayor que el otro, ya que el área al que apunta la cámara está pintada de balnco para no interferir en la umbralización. A continuación, se muestra un ejemplo de la salida del monitor serie mientras se realiza el proceso de umbralización (figura 34).

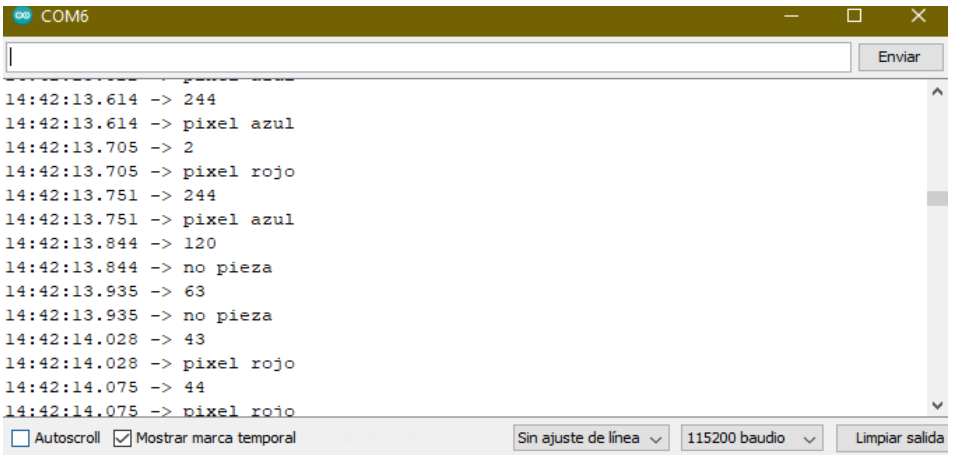


Figura 34. Imagen tomada al monitor serie de arduino que representa el proceso de umbralizado de los pixeles de una imagen

1.6.3.5. Programación de la etapa de labeling

El labeling es un proceso que consiste en asignar un número, la etiqueta, a todos los píxeles contiguos de una forma, para conseguir distinguir esa forma del fondo de la imagen. En este caso, lo que se persigue es conseguir etiquetar todos los objetos del color de interés, no sólo para delimitar su posición sino para poder filtrarlos por tamaño. Seleccionando sólo los más grande, que se corresponderá con la pieza a seleccionar y pudiendo dejar de lado todos aquellos producidos por ruidos.

Esto se ha implementado tomando como referencia el código de Torben Trinkaer Nielsen [19], el cual, recorre la imagen binaria comprobando pixel por pixel si los píxeles que se encuentran a su alrededor pertenecen a la figura o por el contrario al contorno, y si pertenecen a la figura etiqutendolos con un numero nuevo en caso de ser el primero o con el número que ya tiene los pixeles que se encuentran a su alrededor, de esta manera consiguiendo el efecto que deseando. Esto queda ejemplificado en la siguiente imagen (figura 35).

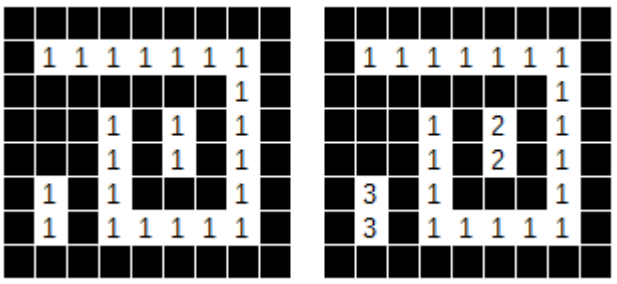


Figura 35. Imagen representativa de un proceso de labeling en una imagen binaria [19]

Pese a que teóricamente el código y su funcionamiento es correcto, el introducir estos algoritmos en el código, y para ello ser necesario tener que definirse un nuevo vector de vectores para el resultado de este proceso, se genera un problema de hardware. Debido a la poca memoria de la que dispone la ESP32-CAM, la implementación de este código aunque se haga mediante memoria dinámica produce un desbordamiento del stack, como consecuencia de las constantes llamadas que hace el código a la función de labeling.

Cabe mencionar la referencia utilizada para desarrollar este código, también presenta en su desarrollo este problema y propone una solución alternativa. Pese a ello, la solución alternativa presentada tampoco es compatible con el dispositivo de cámara, de nuevo debido a esas limitaciones de memoria.

Adicionalmente, buscando una solución a este problema se concluye que, debido a que los errores nacen del llamado reiterado a funciones se puede implementar una versión secuencial del código que permita eliminar las llamadas a estas funciones y por lo tanto, evitando el desbordamiento. Alguno ejemplo de este tipo de proceso de labeling puede ser el proyecto de Michel nicsht [23].

1.6.3.6. Programación de la etapa de cálculo del centro de gravedad

Tras la necesidad de descartar el proceso de labeling como este para el procesado de la imagen, se procederá al cálculo del centro de gravedad. Este proceso consistira en calcular la posición media de todos los pixeles pertenecientes a un color, por un lado su cordeanda x y por otro su cordenada y, de esta manera se obtendra el valor medio de x e y de los pixeles de color, que teoricamente y volienvdo a la hipotesis de que la camra solo detectara como pixeles de color aquellso perenecioentes al objeto a detectar, se obtendria las cordenadas x e y del centro de gravedad. Como se puede apreciar en el siguiente fragmento de código (figura 36).

```
for (unsigned short y = 0; y < SOURCE_HEIGHT-1; y++)
{
  for (unsigned short x = 0; x < SOURCE_WIDTH-1; x++)
  {

   if(umbColumnaA[y] [x]==255){
    auxiliarx = auxiliarx + x;
    auxiliary = auxiliarx + y;
    centroidex = auxiliarx/contadorA;
    centroidey = auxiliary/contadorA;
}
```

Figura 36. Fragmento del código utilizado para recorrer la matriz umbralizada y calcular el centro de gravedad . Extraído del código del anexo A-5

A continuación, sólo sería necesario transformar esas coordenadas de pixeles a milímetros, esto se haría conociendo el tamaño en milímetros de la imagen y realizando una proporción.

Este planteamiento teórico es correcto, pero una vez planteado de forma experimental aparece un problema que invalida su funcionamiento. Este problema es el ruido de la imagen.

Tras la programación de y la puesta a prueba del proyecto, se detecta que los valores de x e y que presenta el sistema son erráticos y muy variables con el tiempo. Esto no debería ser así mientras que la imagen tomada por la cámara permanecerá constante.

Por lo tanto se empieza un proceso de análisis para buscar el causante de dicho error. Tras ese estudio, se puede determinar que la erraticidad de los cálculos viene dada por el ruido que genera la cámara al tomar la imagen. A continuación, se presenta una imagen real tomada con la ESP 32-CAM que permite mostrar las líneas de ruido generadas por la propia cámara (figura 37).

Figura 37. Imagen real tomada por la ESP32-CAM, que permite hacer visible el problema de ruido estático que sufre la cámara.

Como se puede apreciar en la imagen, la cámara proboca una serie de franjas horizontales de color y tamaño aleaorio, estas franjas son las causantes de hacer inestables las medidas, ya que en muchas ocasiones hacen aparecer pixeles rojo o azules.

Esto puede solucionarse con la implementación de un filtro, que antes de comenzar el procesado de la imagen, eliminando las franjas. Con esto se conseguiría poder calcular el centro de gravedad de la imagen utilizando el código ya desarrollado.

1.6.3.7. Conexión serie de la ESP 32-CAM y la placa Arduino Nano

Para poder conseguir un sistema completamente embebido es necesario conseguir comunicar la placa Arduino y la ESP 32-CAM a través del puerto serie pero sin estar conectados al ordenador.

Ambos dispositivos cuentan con un par de pines respectivamente, para el envío y la lectura de datos del puerto serie. En el caso del arduino nano son el pin 0, para la lectura y el pin 1 para el envío, marcado el cero con TX1 y el uno con RX1.

Por lo que respecta a la ESP 32-CAM, sus pines para conexión serie, se corresponden con en pin GPIO 1 para TX0, es decir el de lectura y GPIO 3 para RX0, es decir el envío.

Una vez identificados estos pines será necesario conectarlos de manera cruzada, es decir el pin TX0 de la ESP32-CAM con el pin RX1 del arduino y lo mismo con los dos pines restantes. De esta manera estamos conectando el pin de lectura de un dispositivo con el de envío y viceversa. Utilizando cables dupont.

Con esto se consigue la conexión serie sin necesitar el ordenador, pero en este punto surge un problema. Hasta este momento se han utilizado los puertos USB de ambos dispositivos tanto para la comunicación serie como para alimentarlos.

En este punto y con el objetivo de evitar interferencias en la comunicación serie, será necesario alimentar ambos dispositivos sin el cable USB. En el caso del arduino es fácil, ya que al estar montado sobre el shield se le conectara un transformador a la clavija de jack del arduino nano. Asegurándose que no supere los 5 V ni 1 A de corriente.

En el caso de la ESP 32-CAM, se alimentará contentando a los pines de alimentación del Arduino. Se conectará el pin de 5V del arduino con el de la cámara y se hará lo mismo con el pin GND de la placa de arduino y la cámara, empleando de nuevo cables dupont. Estas conexiones entre el módulo de arduino y la ESP 32-CAM quedan ejemplificadas mediante el siguiente diagrama de conexión (figura 38).

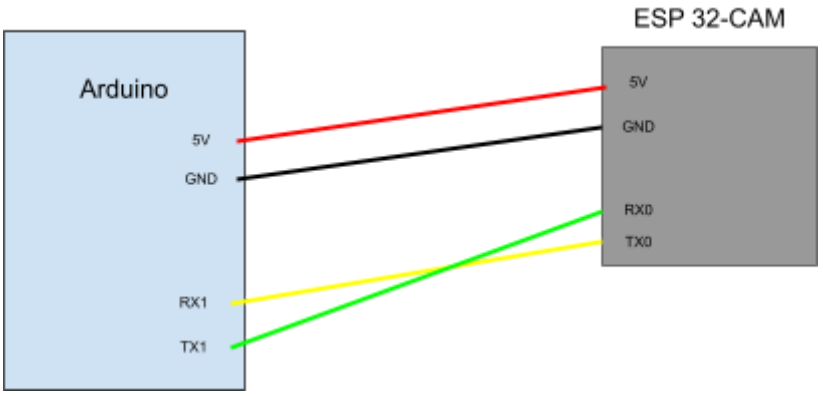


Figura 38. Diagrama de conexión serie y alimentación entre Arduino y ESP 32-CAM

Con estas conexiones realizadas los dos dispositivos se pueden comunicar a través del puerto serie y alimentar, sin necesidad de conectarse a un ordenador. Permitiendo el funcionamiento del sistema de manera embebida completamente.

1.7. TRABAJO RESTANTE Y FUTUROS DESARROLLOS

Como se ha podido apreciar en los últimos apartados del proyecto, no se ha podido implementar las soluciones a los distintos problemas que han surgido durante el desarrollo. Pese a ello si se han encontrado soluciones viables para ambos problemas.

Por lo tanto, para futuras revisiones del proyecto, sería necesario, basándose en un código similar al publicado por Michel nicsht [23], desarrollar en c un sistema de labeling secuencial, que no requiera de las consecutivas llamadas a funciones

Además sería necesario probar y programar esa etapa de filtrado necesaria para eliminar el problema de ruido surgido en el apartado de cálculo del centro de gravedad, por supuesto, con los pertinentes cambios es el resto de etapas para conseguir un correcto funcionamiento de todas con la imagen filtrada.

También, otro planteamiento sería no centrarse en software sino en el hardware, es decir, replantearse la cámara a utilizar, optando esta vez por un modelo aunque quizás más caro, si que tenga las características de memoria y calidad de imagen necesarias para hacer funcionar el código desarrollado.

1.8. CONCLUSIÓN

A modo de conclusión del proyecto, se puede afirmar que aunque no en su totalidad, si se han alcanzado gran parte de los objetivos que se plantearon para el mismo de una manera más que satisfactoria.

Por un lado, sí se ha conseguido un sistema de control de un robot mediante la cámara ESP32-CAM, pero no embebida. Pese a ello sí que se ha conseguido demostrar la viabilidad del proyecto e incluso realizar un procesado parcial de la imagen.

Además, se han planteado las vías de estudio necesarias para conseguir el funcionamiento completo del sistema embebido, aunque por limitaciones de tiempo esas soluciones quedan pendientes de estudio e implementación en una futura revisión del proyecto.

Pero, sobre todo, se ha conseguido realizar un proyecto que permita conocer los fundamentos de la programación de algoritmos para visión artificial, tanto a alto nivel como a bajo. Resultando en una actividad fácilmente replicable que pueda ser de interés para el estudio de futuros alumnos.

1.9. BIBLIOGRAFÍA

[1] novedadesautomatizacion. Imagen robot scara [Consulta: 18 mayo 2022]. Disponible en:
https://novedadesautomatizacion.com/scara/

[2] campetella. Imagen robot antropomórfico [Consulta: 18 mayo 2022]. Disponible en:
https://www.campetella.com/es/industrial-robots/

[3] campetella. Imagen robot cartesiano [Consulta: 18 mayo 2022]. Disponible en:
https://bfmx.com/automatizacion/tipos-de-robots-industriales-mas-utilizados/attachment/robot-cartesiano/

[4] campetella. Imagen robot paralelo [Consulta: 18 mayo 2022]. Disponible en:
https://www.weiss-world.com/es-es/products/robots-10327/robots-delta-211

[5] Venco [Consulta: 14 Abril 2022]. Disponible en:
https://www.vencoel.com/inteligencia-aplicada-a-sistemas-de-vision-artificial-evolucion-y-hardware/

[6] Infaimon [Consulta: 18 Abril 2022]. Disponible en:
https://blog.infaimon.com/historia-vision-artificial/

[7] Doménech Jara, Jorge. (2020). Diseño, desarrollo y programación de un brazo robot de 6 grados de libertad. Utilización en aplicaciones de pick&place. Universitat Politècnica de València. Escuela Técnica Superior de Ingeniería del Diseño - Escola Tècnica Superior d'Enginyeria del Disseny. Recuperdo a partir de:
https://riunet.upv.es/handle/10251/150374

[8] Berjón Valles, Ana. (2019). INTEGRACIÓN DE UN SISTEMA DE VISIÓN ARTIFICIAL EN LA MANO DE UN ROBOT INDUSTRIAL. ICAI – Universidad Pontificia Comillas. Recuperado a partir de:
https://repositorio.comillas.edu/xmlui/bitstream/handle/11531/30667/TFG-%20Berjon%20Valles%2C%20Ana.pdf?sequence=1&isAllowed=y

[9] Baixas Durbán, Rocío. (2020). CREACIÓN DE UN ENTORNO INTERACTIVO MEDIANTE ROBOTS PARA LA CONCIENCIACIÓN DE AYUDA A LA DISCAPACIDAD. ICAI – Universidad Pontificia Comillas. Recuperado a partir de:
https://repositorio.comillas.edu/xmlui/handle/11531/42564

[10] Amazon. Cubos de madera. [Consulta: 8 mayo 2022]. Disponible en:
https://www.amazon.es/Belle-Vous-Cubos-Madera-Pack/dp/B082FXYTPF/ref=sr_1_6?keywords=cubos+madera&qid=1651997797&sr=8-6

[11] Amazon. Cubos de poliestireno expandido.[Consulta: 8 mayo 2022]. Disponible en:
https://www.amazon.es/Poliestireno-Extruido-Blanco-Corcho-Espuma/dp/B0832HHXNN

[12] Amazon. Webcam hama c-400.[Consulta: 8 mayo 2022]. Disponible en:
https://www.amazon.es/Hama-micr%C3%B3fono-Videojuegos-resoluci%C3%B3n-inclinaci%C3%B3n/dp/B08NCJ814H

[13] Amazon. Cámara ESP 32-CAM.[Consulta: 8 mayo 2022]. Disponible en: https://www.amazon.es/ESP32-CAM-ESP32-M%C3%B3dulo-c%C3%A1mara-Bluetooth/dp/B07RT5SC54

[14] Amazon. Cámara Raspberry Pi Camera .[Consulta: 8 mayo 2022]. Disponible en: https://www.amazon.es/C%C3%A1mara-para-Raspberry-Pi-5MP/dp/B09JC6V2XL

[15] Amazon. Plancha de aluminio .[Consulta: 8 mayo 2022]. Disponible en: https://www.amazon.es/Chapa-aluminio-revestida-chapa-elegir/dp/B07ZRX22N1?th=1

[16] Amazon. Plancha de contrachapado .[Consulta: 8 mayo 2022]. Disponible en: https://www.amazon.es/Chely-Siglo-contrachapado-400x600x4-Pirograbado/dp/B07RMB6QLQ

[17] omes-va. *Imagen de espectro de colores para HSV.* [Consulta: 17 mayo 2022]. Disponible en: https://omes-va.com/deteccion-de-colores2/

[18] ruinelli.ch. Función para la conversión de RGB a HSV, publicado a nombre de George Ruinelli el 26 de diciembre de 2010 [Consulta:25 mayo 2022]. Disponible en: https://www.ruinelli.ch/rgb-to-hsv

[19] code project. Función para el labeling de imágenes binarias , publicado a nombre de Torben Trinkaer Nielsen el 1 de diciembre de 2014 [Consulta:30 mayo 2022]. Disponible en: https://www.codeproject.com/Articles/825200/An-Implementation-Of-The-Connected-Component-Label

[20] Javad Ghofroni, «*et al.*». (2019).Machine Vision in the Context of Robotics: A Systematic Literature Review - Dresden, Alemania. [Consulta:5 junio 2022]. Disponible en: https://arxiv.org/pdf/1905.03708.pdf

[21] Massimo Bertozzi, «*et al.*». (2022).Artificial Vision in Road Vehicles. [Consulta:5 junio 2022]. Disponible en: https://d1wqtxts1xzle7.cloudfront.net/42601175/Artificial_vision_in_road_vehicles2016021

[22] Hui Xu. (2022)Robot vision system based on information visualization algorithm. [Consulta: 5 junio 2022]. Disponible en: https://link.springer.com/article/10.1007/s13198-021-01515-y

[23] Michael nischt (2012). función secuencial de laberling de imagenes. [Consulta:5 junio 2022]. Disponible en: https://gist.github.com/michael-nischt/3138545

[24] Simone Salero (2022). Biblioteca para arduino “eloquetvisionArduino” [Consulta:5 junio 2022]. Disponible en: https://github.com/eloquentarduino/EloquentArduino

UNIVERSITAT POLITÈCNICA DE VALÈNCIA

Escuela Técnica Superior de Ingeniería del Diseño

ANEXOS

DISEÑO Y PROGRAMACIÓN DE UN SISTEMA DE BAJO COSTE BASADO EN LA ESP 32-CAM PARA EL CONTROL DE UN BRAZO ROBÓTICO

TRABAJO FINAL DEL

Grado en Ingeniería Electrónica Industrial y Automática

REALIZADO POR

Juan José Martín Osuna

TUTORIZADO POR

Leopoldo Armesto Ángel

Ángel Rodas Jordá

CURSO ACADÉMICO: 2021/2022

UNIVERSITAT POLITÈCNICA DE VALÈNCIA

Escuela Técnica Superior de Ingeniería del Diseño

ANEXO A-1: Código Arduino de partida para el control del brazo robot

DISEÑO Y PROGRAMACIÓN DE UN SISTEMA DE BAJO COSTE BASADO EN LA ESP 32-CAM PARA EL CONTROL DE UN BRAZO ROBÓTICO

TRABAJO FINAL DEL

Grado en Ingeniería Electrónica Industrial y Automática

REALIZADO POR

Juan José Martín Osuna

TUTORIZADO POR

Leopoldo Armesto Ángel

Ángel Rodas Jordá

CURSO ACADÉMICO: 2021/2022

```
//#define ROBOT_EMULATION //Uncomment to use the emulated version in CoppeliaSim (does not move the real robot)
//#define PCA9685  //Uncomment to use the PCA9685 servo driver instead of the Servo library

#ifdef PCA9685
  #include <Adafruit_PWMServoDriver.h>
  Adafruit_PWMServoDriver pwm = Adafruit_PWMServoDriver();
  #define MIN_PWM    130
  #define MAX_PWM   570
#else
  #include <Servo.h>
  Servo servos[12]; //Max number of digital signals
#endif

#define JOINTS  3
typedef struct
{
    uint8_t pin;
    int offset;
    int min_pos;
    int max_pos;
} RobotServo_t;

typedef struct
{
  double L0;
  double H1;
  double L1;
  double L2;
  double L3;
  double L3I;
  double L3O;
  double L4;
  double L5;
  double D1;
  double D2;
  double D5;
}RobotParams_t;



RobotParams_t RobotParams ={5.0,64.5,15.1,80.0,80.0,23.9,35.0,80.0,52.0,75.0,83.7,5.0};//son propios del robot vienen
dados por su geometria

typedef struct
{
double X;
double Y;
double Z;
}RobotPosition_t;

 double qinv[JOINTS];//ANGULOS PARA LA CINEMATICA  INVERSA
RobotPosition_t targetdir;//posición PARA LA CINEMATICA  DIRECTA
```

```
#ifdef ROBOT_EMULATION
RobotServo_t robotServos[JOINTS]={{1,0, 0,180},{2,0,50, 170},{3,0,20,160}};
RobotServo_t pinza={4,0,70, 90};
#else
RobotServo_t robotServos[JOINTS]={{7,-5, 0,180},{6,+10,50, 145},{5,-3,20,160}};  // AQUI SE MODIFICA EL MAXIMO Y EL MINIMO Y EL OFFSET DE LOS SERVOS REALES (PIN AL QUE ESTA CONECTADO, OFFSET, MIN POS, MAX POS)// el 6 mueve la rticulacion fina y el 5 la centarl
RobotServo_t pinza={4,-10,60, 90};//siendo 70º abierto y 90º cerrado
#endif


//inicio definición de posiciones y angulos para desplazarse
double q0[JOINTS]={90.0,90.0,90.0};//ASI SE DEFINE UNA CONFIGURACIÓN


RobotPosition_t base={152.08,-5,146.16};



//fin definición de posiciones y ángulos para desplazarse


//TODO: Define several configurations


void moveAbsJ(const RobotServo_t servos[JOINTS], const double q0[JOINTS], const double qT[JOINTS], const double T); //GENERAR INSTRUCCIÓN DE MOVIMIENTO, ( CONGIGURACIÓN DE LOS SERVOS, PUNTO DEL QUE PARTO Q0, PUNTO AL QUE QUIERO IR Qt, TIEMPO QUE QUIERO TARDAR)
void moveJ(const RobotServo_t robotServos[JOINTS], const double q0[JOINTS], const RobotPosition_t target, const float T, const RobotParams_t Params);// función para mover aplicando la cinematica inversa
void inverseKin(const RobotPosition_t target, const RobotParams_t Params, double qinv[JOINTS]); //FUNCIÓN PARA Aplicar la cinamtica inversa, q* devulve los angulos


void moveL(const RobotServo_t robotServos[JOINTS], const float q0[JOINTS], const RobotPosition_t target, const float T, const RobotParams_t params);//función para mover aplicando la cinematica directa
void forwardKin(const double q[JOINTS], const RobotParams_t params, RobotPosition_t *target);//función para aplicar la cinematica directa


void setup() {
  Serial.begin(115200);
  #ifndef ROBOT_EMULATION
    #ifdef PCA9685
      //Initializes the PCA9685 servo driver
      Wire.pins(0,2);
      Wire.begin(0,2);
      pwm.begin();
      pwm.setPWMFreq(50);
    #endif
  #endif


  //Sets the servo to the initial position
  for (int i=0;i<JOINTS;i++)
    writeServo(robotServos[i],(int)q0[i]);
  delay(3000);


  //TODO: Call moveAbsJ to do some movements


  for (int i=0;i<JOINTS;i++)
    detachServo(robotServos[i]);
}
```

```
void loop() {

}

void writeServo(const RobotServo_t &servo, int angle)
{
  angle=constrain(angle+servo.offset,servo.min_pos,servo.max_pos);
  #ifndef ROBOT_EMULATION
    #ifdef PCA9685
      int pulse_width;
      pulse_width   = map(angle+servo.offset, 0, 180, MIN_PWM, MAX_PWM);
      pwm.setPWM(servo.pin,0,pulse_width);
    #else
      servos[servo.pin].attach(servo.pin);
      servos[servo.pin].write(angle);
    #endif
  #else
    Serial.print("S");
    Serial.print(servo.pin);
    Serial.print(":");
    Serial.print((int)angle,DEC);
    Serial.println(";");
  #endif
}

void detachServo(const RobotServo_t &servo)
{
  #ifndef ROBOT_EMULATION
    #ifdef PCA9685
      pwm.setPWM(servo.pin,0,0);
    #else
      servos[servo.pin].detach();
    #endif
  #endif
}

void moveAbsJ(const RobotServo_t robotServos[JOINTS], const double q0[JOINTS], const double qT[JOINTS], const
double T)
{
  double a[JOINTS],b[JOINTS],c[JOINTS],d[JOINTS],q,t,t0;
  //TODO: Compute trajectory parameters for each joint
  for (int i=0;i<JOINTS;i++)
  {
    a[i]=-((2.0*(qT[i]-q0[i]))/(T*T*T));
    b[i]=((3.0*(qT[i]-q0[i]))/(T*T));
    c[i]=0;
    d[i]=q0[i];

  }
  t0=millis()/1000.0;
  t=0.0;
  while(t<T)
  {
```

```
    for (int i=0;i<JOINTS;i++)
    {
      q=a[i]*(t*t*t)+b[i]*(t*t)+c[i]*t+d[i];
      //TODO: Evaluate the trajectory for joint "i" at time "t" and store the result in "q".
      writeServo(robotServos[i],(int)q);
    }
    delay(20);
    t=millis()/1000.0-t0;
  }
  for (int i=0;i<JOINTS;i++)
    {
      q=qT[i];

      writeServo(robotServos[i],(int)q);
    }
    delay(20);
}

void moveJ(const RobotServo_t robotServos[JOINTS], const double q0[JOINTS], const RobotPosition_t target, const
float T, const RobotParams_t Params)
{

  inverseKin(target, Params, qinv);
  moveAbsJ(robotServos, q0, qinv, T );

}

void inverseKin(const RobotPosition_t target, const RobotParams_t Params, double qinv[JOINTS])
{

  double px=target.X;
  double py=target.Y;
  double pz=target.Z;

  double L0=Params.L0;
  double H1=Params.H1;
  double L1=Params.L1;
  double L2=Params.L2;
  double L3=Params.L3;
  double L3I=Params.L3I;
  double L3O=Params.L3O;
  double L4=Params.L4;
  double L5=Params.L5;
  double D1=Params.D1;
  double D2=Params.D2;
  double D5=Params.D5;

  double P0w [JOINTS],r,ze,s,alpha,betha,gama,q3o,e,psi,fi,arcosseg;

  double q1rad,q2rad,q3rad;

  q1rad=atan2((px-L0),(-py))+asin((D5)/(sqrt((px-L0)*(px-L0)+(py*py))));// CALCULO DEL PRIMER ÁNGULO
```

```
qinv[0]=((q1rad*180.0)/PI); //pasar el angulo 1 a grados


P0w[0]=px+(-L5*sin(q1rad)-L0);// componente x del punto de muñeca

P0w[1]=py+(L5*cos(q1rad));// componente y del punto de muñeca

P0w[2]=pz;// componente z del punto de muñeca


r=sqrt((P0w[0]*P0w[0])+(P0w[1]*P0w[1]))-L1;
ze=P0w[2]-H1;
s=sqrt((r*r)+(ze*ze));

alpha=atan2(ze,r);


arcosseg=(s*s+L2*L2-L3*L3)/(2*s*L2);

 if( arcosseg>=1){
   arcosseg=1;
 }
    if( arcosseg<=-1){
   arcosseg=-1;
 }

betha=acos(arcosseg);

 arcosseg=(L2*L2+L3*L3-s*s)/(2*L2*L3);

 if( arcosseg>=1){
   arcosseg=1;
 }
    if( arcosseg<=-1){
   arcosseg=-1;
 }

gama=acos(arcosseg);

q2rad=PI-alpha-betha;
qinv[1]=((q2rad*180.0)/PI); //pasar el angulo 2 a grados;

q3o=PI-gama;
e=sqrt(L3O*L3O+L2*L2-2*L3O*L2*cos(q3o));
psi=asin((L3O*sin(q3o))/(e));

arcosseg=(e*e+L3I*L3I-L4*L4)/(2*e*L3I);

 if( arcosseg>=1){
   arcosseg=1;
 }
    if( arcosseg<=-1){
   arcosseg=-1;
```

```
    }


  fi=acos(arcosseg);

  q3rad=psi+fi+((PI)/(2.0))-q2rad;
  qinv[2]=((q3rad*180.0)/PI); //pasar el angulo 3 a grados;
}

void moveL(const RobotServo_t robotServos[JOINTS], const double q0[JOINTS], const RobotPosition_t target, const
float T, const RobotParams_t Params)
{

  double P2P, a[JOINTS],b[JOINTS],c[JOINTS],d[JOINTS],s,p,t,t0,pvect[JOINTS];
  RobotPosition_t result;

    forwardKin(q0, Params, &targetdir);



P2P=sqrt((target.X-targetdir.X)*(target.X-targetdir.X)+(target.Y-targetdir.Y)*(target.Y-targetdir.Y)+(target.Z-targetdir.Z)*(ta
rget.Z-targetdir.Z)); // Calcular distancia Euclidiana



     for (int i=0;i<JOINTS;i++)
  {
    a[i]=-((2.0*P2P)/(T*T*T));
    b[i]=((3.0*P2P)/(T*T));
    c[i]=0;
    d[i]=0;

  }

    pvect[0]=((target.X-targetdir.X)/(P2P));
    pvect[1]=((target.Y-targetdir.Y)/(P2P));
    pvect[2]=((target.Z-targetdir.Z)/(P2P));

  t0=millis()/1000.0;
  t=0.0;


  while(t<T)
  {
    for (int i=0;i<JOINTS;i++)
    {
      s=a[i]*(t*t*t)+b[i]*(t*t)+c[i]*t+d[i];
      //TODO: Evaluate the trajectory for joint "i" at time "t" and store the result in "q".

    }
    delay(20);
    t=millis()/1000.0-t0;


      result.X=s*pvect[0]+targetdir.X;
```

```
    result.Y=s*pvect[1]+targetdir.Y;
    result.Z=s*pvect[2]+targetdir.Z;

  inverseKin(result, Params, qinv);

   for (int i=0;i<JOINTS;i++)
  {
    writeServo(robotServos[i],qinv[i]);


  }
   delay(20);
  }}


void forwardKin(const double q[JOINTS], const RobotParams_t Params, RobotPosition_t *target)
{
  double L0=Params.L0;
  double H1=Params.H1;
  double L1=Params.L1;
  double L2=Params.L2;
  double L3=Params.L3;
  double L3I=Params.L3I;
  double L3O=Params.L3O;
  double L4=Params.L4;
  double L5=Params.L5;
  double D1=Params.D1;
  double D2=Params.D2;
  double D5=Params.D5;

  double q1=((q[0]*PI)/180);

  double q2=((q[1]*PI)/180);

  double q3=((q[2]*PI)/180);


  double r, ptx, pty,ptz,q3o,sigma,f,mu,tao,arcosseg;

  sigma=q3+q2-(PI/2.0);


  f=sqrt(L3I*L3I+L2*L2-2.0*L3I*L2*cos(sigma));

   arcosseg=(f*f+L3O*L3O-L4*L4)/(2*f*L3O);

   if( arcosseg>=1){
    arcosseg=1;
   }
      if( arcosseg<=-1){
    arcosseg=-1;
   }
```

```
  mu=acos(arcosseg);


  tao=asin((L3I*sin(sigma))/f);

  q3o=tao+mu;

  r=L1-L2*cos(q2)-L3*cos(q2+q3o);

  target->Z=H1+L2*sin(q2)+L3*sin(q2+q3o);

  target->Y=(-r-L5)*cos(q1)-D5*sin(q1);

  target->X=L0+(r+L5)*sin(q1)-D5*cos(q1);
}
```

UNIVERSITAT POLITÈCNICA DE VALÈNCIA

Escuela Técnica Superior de Ingeniería del Diseño

ANEXO A-2: Código Python para webcam

DISEÑO Y PROGRAMACIÓN DE UN SISTEMA DE BAJO COSTE BASADO EN LA ESP 32-CAM PARA EL CONTROL DE UN BRAZO ROBÓTICO

TRABAJO FINAL DEL

Grado en Ingeniería Electrónica Industrial y Automática

REALIZADO POR

Juan José Martín Osuna

TUTORIZADO POR

Leopoldo Armesto Ángel

Ángel Rodas Jordá

CURSO ACADÉMICO: 2021/2022

```
#-------------------------Declaramos las librerías-------------------------
import cv2
import numpy as np
import serial


#--------------------definimos la fuente de texto-------------------------
font = cv2.FONT_HERSHEY_SIMPLEX

#-------------------------Conexión Arduino----------------------------------
serialArduino = serial.Serial("COM6", 9600)

#time.sleep(10)

#-------------------------Definimos los rangos de detección----------------
azulBajo = np.array([100, 100, 20], np.uint8)
azulAlto = np.array([125, 255, 255], np.uint8)

redBajo1 = np.array([0, 100, 20], np.uint8)
redAlto1 = np.array([10, 255, 255], np.uint8)

redBajo2 =np.array([170, 100, 20], np.uint8)
redAlto2=np.array([180, 255, 255], np.uint8)
# -------------------------Capturas-----------------------------------------
cap = cv2.VideoCapture(0, cv2.CAP_DSHOW)


#-------------------------Calibrado por defecto----------------------------
Xmax = 236
Xmin = 176

Ymax = 520
Ymin = 152
#----------------------------lectura de la imagen----------------------------
while True:

    yA = 3000
    yR = 3000
    xA = 3000
    xR = 3000
    ret, frame = cap.read()

# ----------------------correción de color---------------------------------
    HSV = cv2.cvtColor(frame, cv2.COLOR_BGR2HSV)
# ----------------------Deteción de color en la imagen--------------------
    maskAzul = cv2.inRange(HSV, azulBajo, azulAlto)

    maskRed1 = cv2.inRange(HSV, redBajo1, redAlto1)
    maskRed2 = cv2.inRange(HSV, redBajo2, redAlto2)
    maskRed = cv2.add(maskRed1, maskRed2)
#------------------ Detectar el contorno del color-----------------------
    contornoazul, _ = cv2.findContours(maskAzul, cv2.RETR_EXTERNAL, cv2.CHAIN_APPROX_SIMPLE)
```

```
    contornorojo, _ = cv2.findContours(maskRed, cv2.RETR_EXTERNAL, cv2.CHAIN_APPROX_SIMPLE)

# ----------------detectar el contorno de interés----------------------
    for ca in contornoazul:
        areaazul = cv2.contourArea(ca)
        if areaazul > 3000:
            MA = cv2.moments(ca)
            if(MA["m00"] == 0): MA["m00"] = 1 # Sistema de seguridad en caso de división por cero

            xA = int(MA["m10"]/MA["m00"])
            yA = int(MA["m01"] / MA["m00"])

            cv2.circle(frame,(xA,yA),7,(0,255,0),-1) # Dibjuamos el circulo en la posición central
            cv2.putText(frame,"azul",(xA+10,yA), font, 0.75, (0,255,0),1,cv2.LINE_AA)
            cv2.putText(frame, '{}//{}'.format(xA, yA), (xA - 120, yA -50), font, 0.75, (0, 255, 0), 1, cv2.LINE_AA)

            # funcion para evitar los picos en los contronos
            nuevoContornoAzul = cv2.convexHull(ca)
            # Dibujar contorno
            cv2.drawContours(frame, [nuevoContornoAzul], 0, (255, 0, 0), 3)

    for cr in contornorojo:
        arearoja = cv2.contourArea(cr)
        if arearoja > 3000:
            MR = cv2.moments(cr)
            if (MR["m00"] == 0): MR["m00"] = 1  # Sistema de seguridad en caso de división por cero
            xR = int(MR["m10"] / MR["m00"])
            yR = int(MR["m01"] / MR["m00"])
            cv2.circle(frame, (xR, yR), 7, (0, 255, 0), -1)  # Dibjuamos el circulo en la posición central
            cv2.putText(frame, "rojo", (xR + 10, yR), font, 0.75, (0, 255, 0), 1, cv2.LINE_AA)
            cv2.putText(frame, '{}//{}'.format(xR, yR), (xR + 10, yR - 20), font, 0.75, (0, 255, 0), 1, cv2.LINE_AA)
            #funcion para evitar los picos en los contronos
            nuevoContornoRojo = cv2.convexHull(cr)
            #Dibujar contorno
            cv2.drawContours(frame, [nuevoContornoRojo], 0, (0,0,255), 3)



    cv2.imshow('imagen area', frame)
    t = cv2.waitKey(1)
    if t == 27:
        break
 #--------------------------------Sistema de autocalibrado---------------------------------
    if t == 99:
        Xmin = yA
        Ymin = xA
        print(Xmin)
        print(Ymin)

    if t == 118:
        Xmax = yA
        Ymax = xA
        print(Xmax)
```

```
        print(Ymax)
#-----------------------------Seguridad del calibrado----------------------------------------
    if Xmax <= Xmin | Ymax <= Ymin:
        print("error en el calibrado")
        print("Establecidos parámetros por defecto")

        Xmax = 236
        Xmin = 176

        Ymax = 520
        Ymin = 152

    if t == 32:
        if (yA < yR):
            valorpixely = yA
            valorpixelx = xA
            color = str(0)
        elif (yR < yA):
            valorpixely = yR
            valorpixelx = xR
            color = str(1)
        else:
            break

        print(Xmax)
        print(Xmin)
        print(Ymax)
        print(Ymin)

        valorx = int(60/(Xmax-Xmin)*(valorpixely-Xmin))
        valory = int(152/(Ymax-Ymin)*(valorpixelx-Ymin))

        if valorx < 0:
            valorx = 0
        if valory < 0:
            valory = 0

        if valorx < 100 & valorx >= 10:
            centenax = 0
            decenax = int((valorx - (centenax * 100)) / 10)
            unidadx = int(valorx - (centenax * 100 + decenax * 10))
        elif valorx < 10 & valorx >= 1:
            centenax = 0
            decenax = 0
            unidadx = int(valorx - (centenax * 100 + decenax * 10))
        elif valorx < 1:
            centenax = 0
            decenax = 0
            unidadx = 0
        else:
            centenax = int(valorx / 100)
            decenax = int((valorx - (centenax * 100)) / 10)
            unidadx = int(valorx - (centenax * 100 + decenax * 10))
```

```
        centenax = str(centenax)
        decenax = str(decenax)
        unidadx = str(unidadx)

        if valory < 100 & valory >= 10:
            centenay = 0
            decenay = int((valory - (centenay * 100)) / 10)
            unidady = int(valory - (centenay * 100 + decenay * 10))
        elif valory < 10 & valory >= 1:
            centenay = 0
            decenay = 0
            unidady = int(valory - (centenay * 100 + decenay * 10))
        elif valory < 1:
            centenay = 0
            decenay = 0
            unidady = 0
        else:
            centenay = int(valory / 100)
            decenay = int((valory - (centenay * 100)) / 10)
            unidady = int(valory - (centenay * 100 + decenay * 10))

        centenay = str(centenay)
        decenay = str(decenay)
        unidady = str(unidady)

        print(centenay)
        print(decenay)
        print(unidady)


        serialArduino.write("@".encode('ascii'))
        serialArduino.flush()
        serialArduino.write(centenax.encode('ascii'))
        serialArduino.write(decenax.encode('ascii'))
        serialArduino.write(unidadx.encode('ascii'))
        serialArduino.write(centenay.encode('ascii'))
        serialArduino.write(decenay.encode('ascii'))
        serialArduino.write(unidady.encode('ascii'))
        serialArduino.write(color.encode('ascii'))
        serialArduino.write("*".encode('ascii'))



    if t == 27:
        break
cv2.destroyAllWindows()
```

UNIVERSITAT POLITÈCNICA DE VALÈNCIA

Escuela Técnica Superior de Ingeniería del Diseño

ANEXO A-3: Código Arduino para control del robot a través del monitor serie

DISEÑO Y PROGRAMACIÓN DE UN SISTEMA DE BAJO COSTE BASADO EN LA ESP 32-CAM PARA EL CONTROL DE UN BRAZO ROBÓTICO

TRABAJO FINAL DEL

Grado en Ingeniería Electrónica Industrial y Automática

REALIZADO POR

Juan José Martín Osuna

TUTORIZADO POR

Leopoldo Armesto Ángel

Ángel Rodas Jordá

CURSO ACADÉMICO: 2021/2022

```
/#define ROBOT_EMULATION //Uncomment to use the emulated version in CoppeliaSim (does not move the real robot)
//#define PCA9685  //Uncomment to use the PCA9685 servo driver instead of the Servo library

#ifdef PCA9685
  #include <Adafruit_PWMServoDriver.h>
  Adafruit_PWMServoDriver pwm = Adafruit_PWMServoDriver();
  #define MIN_PWM    130
  #define MAX_PWM   570
#else
  #include <Servo.h>
  Servo servos[12]; //Max number of digital signals
#endif

#define JOINTS  3

int bt_pos = 0;
int color = 0;
unsigned char bt_aux;
unsigned char bt_sig;
typedef struct
{
    uint8_t pin;
    int offset;
    int min_pos;
    int max_pos;
} RobotServo_t;

typedef struct
{
  double L0;
  double H1;
  double L1;
  double L2;
  double L3;
  double L3I;
  double L3O;
  double L4;
  double L5;
  double D1;
  double D2;
  double D5;
}RobotParams_t;



RobotParams_t RobotParams ={5.0,64.5,15.1,80.0,80.0,23.9,35.0,80.0,52.0,75.0,83.7,5.0};//son propios del robot vienen
dados por su geometria

typedef struct
{
double X;
double Y;
```

```
double Z;
}RobotPosition_t;

 double qinv[JOINTS];//ANGULOS PARA LA CINEMATICA INVERSA
RobotPosition_t targetdir;//posición PARA LA CINEMATICA DIRECTA

#ifdef ROBOT_EMULATION
RobotServo_t robotServos[JOINTS]={{1,0, 0,180},{2,0,50, 170},{3,0,20,160}};
RobotServo_t pinza={4,0,70, 90};
#else
RobotServo_t robotServos[JOINTS]={{7,0, 0,180},{6,-10,0 ,180},{5,0,20,160}};  // AQUI SE MODIFICA EL MAXIMO Y EL MINIMO Y EL OFFSET DE LOS SERVOS REALES (PIN AL QUE ESTA CONECTADO, OFFSET, MIN POS, MAX POS)// el 6 mueve la rticulacion fina y el 5 la centarl
RobotServo_t pinza={4,-10,50, 90};//siendo 70º abierto y 90º cerrado
#endif

//inicio definición de posiciones y angulos para desplazarse
double q0[JOINTS]={90.0,60.0,90.0};//ASI SE DEFINE UNA CONFIGURACIÓN
double qrojo[JOINTS]={0.0,160.0,160.0};
double qazul[JOINTS]={20.0,150.0,160.0};
double aux[JOINTS]={90.0,90.0,90.0};

RobotPosition_t base={152.08,-5,146.16};
RobotPosition_t obj={120.00,-80,30.36};//la X se suma, la Y se resta y la Z cosntante (130:190) (80:-100)




//fin definición de posiciones y angulos para desplazarse

//TODO: Define several configurations
void writeServo(const RobotServo_t &servo, int angle);
void detachServo(const RobotServo_t &servo);
void moveAbsJ(const RobotServo_t servos[JOINTS], const double q0[JOINTS], const double qT[JOINTS], const double T); //GENERAR INSTRUCCIÓN DE MOVIMIENTO, ( CONGIGURACIÓN DE LOS SERVOS, PUNTO DEL QUE PARTO Q0, PUNTO AL QUE QUIERO IR Qt, TIEMPO QUE QUIERO TARDAR)
void moveJ(const RobotServo_t robotServos[JOINTS], const double q0[JOINTS], const RobotPosition_t target, const float T, const RobotParams_t Params);// función para mover aplicando la cinematica inversa
void inverseKin(const RobotPosition_t target, const RobotParams_t Params, double qinv[JOINTS]); //FUNCIÓN PARA Aplicar la cinamtica inversa, q* devulve los angulos

void moveL(const RobotServo_t robotServos[JOINTS], const float q0[JOINTS], const RobotPosition_t target, const float T, const RobotParams_t params);//función para mover aplicando la cinematica directa
void forwardKin(const double q[JOINTS], const RobotParams_t params, RobotPosition_t *target);//función para aplicar la cinematica directa
void dejarRojo();
void dejarAzul();

void setup() {
  Serial.begin(9600);
  #ifndef ROBOT_EMULATION
    #ifdef PCA9685
      //Initializes the PCA9685 servo driver
      Wire.pins(0,2);
      Wire.begin(0,2);
      pwm.begin();
```

```
    pwm.setPWMFreq(50);
   #endif
  #endif

  //Sets the servo to the initial position
  for (int i=0;i<JOINTS;i++)
    writeServo(robotServos[i],(int)q0[i]);
    writeServo(pinza, 30);
  delay(3000);

  //TODO: Call moveAbsJ to do some movements

  for (int i=0;i<JOINTS;i++)
    detachServo(robotServos[i]);
}

void loop() {

   while(Serial.available()>=0)
    {

      unsigned char c;
      Serial.readBytes(&c,1);


     if((c =='@') && (bt_pos == 0)){
         bt_pos++;
         Serial.print("Inicio lectura");

         }

     else if ( bt_pos == 1){
        bt_pos++;
         bt_aux = c-48;
         Serial.print("\nValor leido1: ");
         Serial.println(bt_aux);
         obj.X = obj.X +(100*bt_aux);

     }
      else if ( bt_pos == 2){
        bt_pos++;
         bt_aux = c-48;
         Serial.print("\nValor leido2: ");
         Serial.println(bt_aux);
         obj.X = obj.X +(10*bt_aux);
     }
      else if ( bt_pos == 3){
        bt_pos++;
         bt_aux = c-48;
         Serial.print("\nValor leido3: ") ;
         Serial.println(bt_aux);
         obj.X = obj.X +(bt_aux);
          if(bt_sig == 1){
```

```
          obj.X = obj.X * -1;
         }
      }

      else if ( bt_pos == 4){
        bt_pos++;
         bt_aux = c-48;
         Serial.print("\nValor leido1: ");
         Serial.println(bt_aux);
         obj.Y = obj.Y +(100*bt_aux);

      }
       else if ( bt_pos == 5){
        bt_pos++;
         bt_aux = c-48;
         Serial.print("\nValor leido2: ");
         Serial.println(bt_aux);
         obj.Y = obj.Y +(10*bt_aux);
      }
       else if ( bt_pos == 6){
        bt_pos++;
         bt_aux = c-48;
         Serial.print("\nValor leido3: ") ;
         Serial.println(bt_aux);
         obj.Y = obj.Y +(bt_aux);

      }
      else if( bt_pos == 7){
       bt_pos++;
       bt_aux = c - 48;
       color =  bt_aux;
      }
      else if ( bt_pos == 8&&(c=='*')){
        bt_pos=0;
        Serial.println("Inicio despalzamiento");
        writeServo(pinza, 30);
        moveL(robotServos,q0,obj, 3.0, RobotParams);
        if (color ==0 )dejarAzul();
        else dejarRojo();

        obj.Y = -80;
        obj.X = 120;



      }

 }
}

void writeServo(const RobotServo_t &servo, int angle)
{
  angle=constrain(angle+servo.offset,servo.min_pos,servo.max_pos);
```

```
  #ifndef ROBOT_EMULATION
    #ifdef PCA9685
      int pulse_width;
      pulse_width   = map(angle+servo.offset, 0, 180, MIN_PWM, MAX_PWM);
      pwm.setPWM(servo.pin,0,pulse_width);
    #else
      servos[servo.pin].attach(servo.pin);
      servos[servo.pin].write(angle);
    #endif
  #else
    Serial.print("S");
    Serial.print(servo.pin);
    Serial.print(":");
    Serial.print((int)angle,DEC);
    Serial.println(";");
  #endif
}
void dejarRojo()
{
  writeServo(pinza, 90);
  delay(1000);
  inverseKin(obj, RobotParams, aux);
  moveAbsJ(robotServos,aux,q0, 5.0);
  moveAbsJ(robotServos,q0,qrojo, 5.0);
  writeServo(pinza, 70);
  moveAbsJ(robotServos, qrojo,q0, 5.0);
  writeServo(pinza, 90);

}
void dejarAzul()
{
  writeServo(pinza, 90);
   delay(1000);
  inverseKin(obj, RobotParams, aux);
  moveAbsJ(robotServos,aux,q0, 5.0);
  moveAbsJ(robotServos,q0,qazul, 5.0);
  writeServo(pinza, 70);
  moveAbsJ(robotServos, qazul,q0, 5.0);
  writeServo(pinza, 90);

}

void detachServo(const RobotServo_t &servo)
{
  #ifndef ROBOT_EMULATION
    #ifdef PCA9685
      pwm.setPWM(servo.pin,0,0);
    #else
      servos[servo.pin].detach();
    #endif
  #endif
}
```

```
void moveAbsJ(const RobotServo_t robotServos[JOINTS], const double q0[JOINTS], const double qT[JOINTS], const double T)
{
  double a[JOINTS],b[JOINTS],c[JOINTS],d[JOINTS],q,t,t0;
  //TODO: Compute trajectory parameters for each joint
  for (int i=0;i<JOINTS;i++)
  {
    a[i]=-((2.0*(qT[i]-q0[i]))/(T*T*T));
    b[i]=((3.0*(qT[i]-q0[i]))/(T*T));
    c[i]=0;
    d[i]=q0[i];

  }
  t0=millis()/1000.0;
  t=0.0;
  while(t<T)
  {
    for (int i=0;i<JOINTS;i++)
    {
      q=a[i]*(t*t*t)+b[i]*(t*t)+c[i]*t+d[i];
      //TODO: Evaluate the trajectory for joint "i" at time "t" and store the result in "q".
      writeServo(robotServos[i],(int)q);
    }
    delay(20);
    t=millis()/1000.0-t0;
  }
  for (int i=0;i<JOINTS;i++)
    {
      q=qT[i];

      writeServo(robotServos[i],(int)q);
    }
    delay(20);
}

void moveJ(const RobotServo_t robotServos[JOINTS], const double q0[JOINTS], const RobotPosition_t target, const float T, const RobotParams_t Params)
{

  inverseKin(target, Params, qinv);
  moveAbsJ(robotServos, q0, qinv, T );

}

void inverseKin(const RobotPosition_t target, const RobotParams_t Params, double qinv[JOINTS])
{

  double px=target.X;
  double py=target.Y;
  double pz=target.Z;

  double L0=Params.L0;
  double H1=Params.H1;
  double L1=Params.L1;
```

```
double L2=Params.L2;
double L3=Params.L3;
double L3I=Params.L3I;
double L3O=Params.L3O;
double L4=Params.L4;
double L5=Params.L5;
double D1=Params.D1;
double D2=Params.D2;
double D5=Params.D5;

double P0w [JOINTS],r,ze,s,alpha,betha,gama,q3o,e,psi,fi,arcosseg;

double q1rad,q2rad,q3rad;

q1rad=atan2((px-L0),(-py))+asin((D5)/(sqrt((px-L0)*(px-L0)+(py*py))));// CALCULO DEL PRIMER ANGULO

qinv[0]=((q1rad*180.0)/PI); //pasar el angulo 1 a grados


P0w[0]=px+(-L5*sin(q1rad)-L0);// componente x del punto de muñeca

P0w[1]=py+(L5*cos(q1rad));// componente y del punto de muñeca

P0w[2]=pz;// componente z del punto de muñeca


r=sqrt((P0w[0]*P0w[0])+(P0w[1]*P0w[1]))-L1;
ze=P0w[2]-H1;
s=sqrt((r*r)+(ze*ze));

alpha=atan2(ze,r);


arcosseg=(s*s+L2*L2-L3*L3)/(2*s*L2);

 if( arcosseg>=1){
   arcosseg=1;
 }
    if( arcosseg<=-1){
   arcosseg=-1;
 }

betha=acos(arcosseg);

 arcosseg=(L2*L2+L3*L3-s*s)/(2*L2*L3);

 if( arcosseg>=1){
   arcosseg=1;
 }
    if( arcosseg<=-1){
   arcosseg=-1;
 }
```

```
  gama=acos(arcosseg);

  q2rad=PI-alpha-betha;
  qinv[1]=((q2rad*180.0)/PI); //pasar el angulo 2 a grados;

  q3o=PI-gama;
  e=sqrt(L3O*L3O+L2*L2-2*L3O*L2*cos(q3o));
  psi=asin((L3O*sin(q3o))/(e));

  arcosseg=(e*e+L3I*L3I-L4*L4)/(2*e*L3I);

    if( arcosseg>=1){
      arcosseg=1;
    }
        if( arcosseg<=-1){
      arcosseg=-1;
    }


  fi=acos(arcosseg);

  q3rad=psi+fi+((PI)/(2.0))-q2rad;
  qinv[2]=((q3rad*180.0)/PI); //pasar el angulo 3 a grados;
}

void moveL(const RobotServo_t robotServos[JOINTS], const double q0[JOINTS], const RobotPosition_t target, const
float T, const RobotParams_t Params)
{

  double P2P, a[JOINTS],b[JOINTS],c[JOINTS],d[JOINTS],s,p,t,t0,pvect[JOINTS];
  RobotPosition_t result;

    forwardKin(q0, Params, &targetdir);



P2P=sqrt((target.X-targetdir.X)*(target.X-targetdir.X)+(target.Y-targetdir.Y)*(target.Y-targetdir.Y)+(target.Z-targetdir.Z)*(ta
rget.Z-targetdir.Z)); // Calcular distancia Euclidiana



      for (int i=0;i<JOINTS;i++)
  {
    a[i]=-((2.0*P2P)/(T*T*T));
    b[i]=((3.0*P2P)/(T*T));
    c[i]=0;
    d[i]=0;

  }

    pvect[0]=((target.X-targetdir.X)/(P2P));
    pvect[1]=((target.Y-targetdir.Y)/(P2P));
    pvect[2]=((target.Z-targetdir.Z)/(P2P));

  t0=millis()/1000.0;
```

```
t=0.0;


while(t<T)
{
  for (int i=0;i<JOINTS;i++)
  {
    s=a[i]*(t*t*t)+b[i]*(t*t)+c[i]*t+d[i];
    //TODO: Evaluate the trajectory for joint "i" at time "t" and store the result in "q".

  }
  delay(20);
  t=millis()/1000.0-t0;


    result.X=s*pvect[0]+targetdir.X;
    result.Y=s*pvect[1]+targetdir.Y;
    result.Z=s*pvect[2]+targetdir.Z;

 inverseKin(result, Params, qinv);

  for (int i=0;i<JOINTS;i++)
{
    writeServo(robotServos[i],qinv[i]);


}
  delay(20);
}}


void forwardKin(const double q[JOINTS], const RobotParams_t Params, RobotPosition_t *target)
{
  double L0=Params.L0;
  double H1=Params.H1;
  double L1=Params.L1;
  double L2=Params.L2;
  double L3=Params.L3;
  double L3I=Params.L3I;
  double L3O=Params.L3O;
  double L4=Params.L4;
  double L5=Params.L5;
  double D1=Params.D1;
  double D2=Params.D2;
  double D5=Params.D5;

  double q1=((q[0]*PI)/180);

  double q2=((q[1]*PI)/180);

  double q3=((q[2]*PI)/180);
```

```
  double r, ptx, pty,ptz,q3o,sigma,f,mu,tao,arcosseg;

  sigma=q3+q2-(PI/2.0);


  f=sqrt(L3I*L3I+L2*L2-2.0*L3I*L2*cos(sigma));

   arcosseg=(f*f+L3O*L3O-L4*L4)/(2*f*L3O);

   if( arcosseg>=1){
     arcosseg=1;
   }
       if( arcosseg<=-1){
     arcosseg=-1;
   }

  mu=acos(arcosseg);


  tao=asin((L3I*sin(sigma))/f);

  q3o=tao+mu;

  r=L1-L2*cos(q2)-L3*cos(q2+q3o);

  target->Z=H1+L2*sin(q2)+L3*sin(q2+q3o);

  target->Y=(-r-L5)*cos(q1)-D5*sin(q1);

  target->X=L0+(r+L5)*sin(q1)-D5*cos(q1);
}
```

UNIVERSITAT POLITÈCNICA DE VALÈNCIA

Escuela Técnica Superior de Ingeniería del Diseño

ANEXO A-4: Código Python para ESP 32-CAM

DISEÑO Y PROGRAMACIÓN DE UN SISTEMA DE BAJO COSTE BASADO EN LA ESP 32-CAM PARA EL CONTROL DE UN BRAZO ROBÓTICO

TRABAJO FINAL DEL

Grado en Ingeniería Electrónica Industrial y Automática

REALIZADO POR

Juan José Martín Osuna

TUTORIZADO POR

Leopoldo Armesto Ángel

Ángel Rodas Jordá

CURSO ACADÉMICO: 2021/2022

```
#-------------------------Declaramos las librerías-------------------------
  import cv2
  import numpy as np
  import serial
  from urllib.request import urlopen


  #---------------------definimos la fuente de texto-------------------------
  font = cv2.FONT_HERSHEY_SIMPLEX

  #-------------------------Conexión Arduino-----------------------------------
  serialArduino = serial.Serial("COM4", 9600)

  #time.sleep(10)

  #-------------------------Definimos los rangos de detección----------------
  azulBajo = np.array([100, 100, 20], np.uint8)    #  100,20
  azulAlto = np.array([125, 255, 255], np.uint8)    # 255, 255

  redBajo1 = np.array([0, 100, 20], np.uint8)
  redAlto1 = np.array([10, 255, 255], np.uint8)

  redBajo2 =np.array([170, 100, 20], np.uint8)
  redAlto2=np.array([180, 255, 255], np.uint8)
  # -------------------------Capturas-----------------------------------------
  stream = urlopen('http://192.168.93.7:81/stream')
  bytes = bytes()

  #-------------------------Calibrado por defecto-----------------------------
  Xmax = 137
  Xmin = 118

  Ymax = 296
  Ymin = 29

  #-----------------------------lectura de la imagen-----------------------------
  while True:
     bytes += stream.read(1024)
     a = bytes.find(b'\xff\xd8')
     b = bytes.find(b'\xff\xd9')
     if a != -1 and b != -1:
        jpg = bytes[a:b + 2]
        bytes = bytes[b + 2:]
        if jpg:
           frame = cv2.imdecode(np.fromstring(jpg, dtype=np.uint8), cv2.IMREAD_COLOR)



           yA = 3000
           yR = 3000
           xA = 3000
           xR = 3000
```

```
# ----------------------correción de color---------------------------------
        HSV = cv2.cvtColor(frame, cv2.COLOR_BGR2HSV)
# ----------------------Deteción de color en la imagen--------------------
        maskAzul = cv2.inRange(HSV, azulBajo, azulAlto)

        maskRed1 = cv2.inRange(HSV, redBajo1, redAlto1)
        maskRed2 = cv2.inRange(HSV, redBajo2, redAlto2)
        maskRed = cv2.add(maskRed1, maskRed2)
#----------------- Detectar el contorno del color-----------------------
        contornoazul, _ = cv2.findContours(maskAzul, cv2.RETR_EXTERNAL, cv2.CHAIN_APPROX_SIMPLE)

        contornorojo, _ = cv2.findContours(maskRed, cv2.RETR_EXTERNAL, cv2.CHAIN_APPROX_SIMPLE)

# -----------------seleccionar el contorno de interés----------------------
        for ca in contornoazul:
            areaazul = cv2.contourArea(ca)
            if areaazul > 1000:
                MA = cv2.moments(ca)
                if(MA["m00"] == 0): MA["m00"] = 1 # Sistema de seguridad en caso de división por cero

                xA = int(MA["m10"]/MA["m00"])
                yA = int(MA["m01"] / MA["m00"])

                cv2.circle(frame,(xA,yA),7,(0,255,0),-1) # Dibjuamos el circulo en la posición central
                cv2.putText(frame,"azul",(xA+10,yA), font, 0.75, (0,255,0),1,cv2.LINE_AA)
                cv2.putText(frame, '{}//{}'.format(xA, yA), (xA - 100, yA -50), font, 0.75, (0, 255, 0), 1, cv2.LINE_AA)

        # funcion para evitar los picos en los contronos
                nuevoContornoAzul = cv2.convexHull(ca)
        # Dibujar contorno
                cv2.drawContours(frame, [nuevoContornoAzul], 0, (255, 0, 0), 3)

        for cr in contornorojo:
            arearoja = cv2.contourArea(cr)
            if arearoja > 500:
                MR = cv2.moments(cr)
                if (MR["m00"] == 0): MR["m00"] = 1  # Sistema de seguridad en caso de división por cero
                xR = int(MR["m10"] / MR["m00"])
                yR = int(MR["m01"] / MR["m00"])
                cv2.circle(frame, (xR, yR), 7, (0, 255, 0), -1)  # Dibjuamos el circulo en la posición central
                cv2.putText(frame, "rojo", (xR + 10, yR), font, 0.75, (0, 255, 0), 1, cv2.LINE_AA)
                cv2.putText(frame, '{}//{}'.format(xR, yR), (xR + 10, yR - 20), font, 0.75, (0, 255, 0), 1, cv2.LINE_AA)
        #funcion para evitar los picos en los contronos
                nuevoContornoRojo = cv2.convexHull(cr)
        #Dibujar contorno
                cv2.drawContours(frame, [nuevoContornoRojo], 0, (0,0,255), 3)



    cv2.imshow('imagen area', frame)
t = cv2.waitKey(1)
```

```
    if t == 27:
        break
#---------------------------------Sistema de autocalibrado---------------------------------
    if t == 99:
        Xmin = yA
        Ymin = xA
        print(Xmin)
        print(Ymin)

    if t == 118:
        Xmax = yA
        Ymax = xA
        print(Xmax)
        print(Ymax)
#----------------------------Seguridad del calibrado-----------------------------------------
    if Xmax <= Xmin | Ymax <= Ymin:
        print("error en el calibrado")
        print("Establecidos parámetros por defecto")

        Xmax = 137
        Xmin = 118

        Ymax = 296
        Ymin = 29

    if t == 32:
        if (yA < yR):
            valorpixely = yA
            valorpixelx = xA
            color = str(0)
        elif (yR < yA):
            valorpixely = yR
            valorpixelx = xR
            color = str(1)
        else:
            break
        valorx = int(60/(Xmax-Xmin)*(valorpixely-Xmin))
        valory = int(152/(Ymax-Ymin)*(valorpixelx-Ymin))

        if valorx < 0:
            valorx = 0
        if valory < 0:
            valory = 0

        if valorx < 100 & valorx >= 10:
            centenax = 0
            decenax = int((valorx - (centenax * 100)) / 10)
            unidadx = int(valorx - (centenax * 100 + decenax * 10))
        elif valorx < 10 & valorx >= 1:
            centenax = 0
            decenax = 0
            unidadx = int(valorx - (centenax * 100 + decenax * 10))
        elif valorx < 1:
```

```
            centenax = 0
            decenax = 0
            unidadx = 0
        else:
            centenax = int(valorx / 100)
            decenax = int((valorx - (centenax * 100)) / 10)
            unidadx = int(valorx - (centenax * 100 + decenax * 10))

        centenax = str(centenax)
        decenax = str(decenax)
        unidadx = str(unidadx)

        if valory < 100 & valory >= 10:
            centenay = 0
            decenay = int((valory - (centenay * 100)) / 10)
            unidady = int(valory - (centenay * 100 + decenay * 10))
        elif valory < 10 & valory >= 1:
            centenay = 0
            decenay = 0
            unidady = int(valory - (centenay * 100 + decenay * 10))
        elif valory < 1:
            centenay = 0
            decenay = 0
            unidady = 0
        else:
            centenay = int(valory / 100)
            decenay = int((valory - (centenay * 100)) / 10)
            unidady = int(valory - (centenay * 100 + decenay * 10))

        centenay = str(centenay)
        decenay = str(decenay)
        unidady = str(unidady)

        print(centenay)
        print(decenay)
        print(unidady)


        serialArduino.write("@".encode('ascii'))
        serialArduino.flush()
        serialArduino.write(centenax.encode('ascii'))
        serialArduino.write(decenax.encode('ascii'))
        serialArduino.write(unidadx.encode('ascii'))
        serialArduino.write(centenay.encode('ascii'))
        serialArduino.write(decenay.encode('ascii'))
        serialArduino.write(unidady.encode('ascii'))
        serialArduino.write(color.encode('ascii'))
        serialArduino.write("*".encode('ascii'))


    if t == 27:
        break
cv2.destroyAllWindows()
```

UNIVERSITAT POLITÈCNICA DE VALÈNCIA

Escuela Técnica Superior de Ingeniería del Diseño

ANEXO A-5: Código Arduino para el labeling de la imagen embebida en la ESP32-CAM

DISEÑO Y PROGRAMACIÓN DE UN SISTEMA DE BAJO COSTE BASADO EN LA ESP 32-CAM PARA EL CONTROL DE UN BRAZO ROBÓTICO

TRABAJO FINAL DEL

Grado en Ingeniería Electrónica Industrial y Automática

REALIZADO POR

Juan José Martín Osuna

TUTORIZADO POR

Leopoldo Armesto Ángel

Ángel Rodas Jordá

CURSO ACADÉMICO: 2021/2022

```
// Change according to your model
// The models available are
//   - CAMERA_MODEL_WROVER_KIT
//   - CAMERA_MODEL_ESP_EYE
//   - CAMERA_MODEL_M5STACK_PSRAM
//   - CAMERA_MODEL_M5STACK_WIDE
//   - CAMERA_MODEL_AI_THINKER
#define CAMERA_MODEL_AI_THINKER

#include <FS.h>
#include <SPIFFS.h>
#include "EloquentVision.h"


#define FRAME_SIZE FRAMESIZE_QVGA
#define SOURCE_WIDTH 320
#define SOURCE_HEIGHT 240

using namespace Eloquent::Vision;
using namespace Eloquent::Vision::IO;
using namespace Eloquent::Vision::ImageProcessing;

// an easy interface to capture images from the camera
ESP32Camera camera;


uint8_t** umbColumnaR = (uint8_t**)malloc(SOURCE_HEIGHT * sizeof(uint8_t*));
uint8_t** umbColumnaA = (uint8_t**)malloc(SOURCE_HEIGHT * sizeof(uint8_t*));
uint8_t** Labeling = (uint8_t**)malloc(SOURCE_HEIGHT * sizeof(uint8_t*));

//Funcion auxiliar
uint8_t h;
uint32_t contadorR = 0, contadorA = 0;



void setup() {
  Serial.begin(115200);
  camera.begin(FRAME_SIZE, PIXFORMAT_RGB565);


  for ( uint32_t i = 0; i < SOURCE_HEIGHT ; i++) {

     umbColumnaR[i]=(uint8_t*)malloc(SOURCE_WIDTH * sizeof(uint8_t));
     umbColumnaA[i]=(uint8_t*)malloc(SOURCE_WIDTH * sizeof(uint8_t));
     Labeling[i]=(uint8_t*)malloc(SOURCE_WIDTH * sizeof(uint8_t));
  }

  for (unsigned short y = 0; y < SOURCE_HEIGHT; y++)
    for (unsigned short x = 0; x < SOURCE_WIDTH; x++){
       umbColumnaR[y][x] = 0;
       umbColumnaA[y][x] = 0;
       Labeling[y][x] = 0;
```

```
    }

}

  /**

  */
  void loop() {
    uint32_t start = millis();
    camera_fb_t *frame = camera.capture();

     unsigned short desplazarX ;
     unsigned short desplazarY ;
     desplazarX = 0;
     desplazarY = 0;
     contadorR = 0;
     contadorA = 0;


    Serial.println("Foto tomada ");
    for ( uint32_t e = 0; e < (SOURCE_WIDTH * SOURCE_HEIGHT * 2) - 2; e = e + 2) {


     // Serial.println(e);
      uint8_t high = frame->buf[e];
      uint8_t low  = frame->buf[e + 1];
      uint16_t pixel = (high << 8) | low;

      uint8_t r = (pixel & 0b1111100000000000) >> 11;
      uint8_t g = (pixel & 0b0000011111100000) >> 5;
      uint8_t b = (pixel & 0b0000000000011111);

      //Serial.print("Red: ");
     // Serial.print(r);
      //Serial.print(", Green: ");
      //Serial.print(g);
      //Serial.print(", Blue: ");
      //Serial.println(b);

      if(r==0) r=0.000001;
      if(g==0) r=0.000001;
      if(b==0) r=0.000001;

      // redefinición rango de colores
      uint8_t R = (255 * r) / 31;
      uint8_t G = (255 * g) / 63;
      uint8_t B = (255 * b) / 31;
      //pasar a h de hasv
      h = RGB2HSV(R, G, B);

      //Serial.print(desplazarX);
      //Serial.print("-");
```

```
    //Serial.println(desplazarY);

    if ((h > 0 && h < 50) || (h > 170 && h < 180))
    {
      //Serial.print("pixel rojo");
      contadorR++;

      umbColumnaR[desplazarY] [desplazarX] = 255;
      umbColumnaA[desplazarY] [desplazarX] = 0;
    }
    else if (h > 230 && h < 255) {
      //Serial.print("pixel azul");
      contadorA++;
      umbColumnaR[desplazarY] [desplazarX] = 0;
      umbColumnaA[desplazarY] [desplazarX] = 255;
    }
    else
    {
     umbColumnaR[desplazarY] [desplazarX] = 0;
     umbColumnaA[desplazarY] [desplazarX] = 0;

    }



 desplazarX++;

 if( desplazarX == (SOURCE_WIDTH ))
 {
   desplazarX = 0;
   desplazarY++;
// if( desplazarY == (SOURCE_HEIGHT ))
  // {
    // desplazarY = 0;
  // }

 }

 }
    Serial.println("final imagen ");
    if (contadorA>contadorR){
 LabelImage(SOURCE_WIDTH, SOURCE_HEIGHT, umbColumnaA, Labeling);
    }
    else{
 LabelImage(SOURCE_WIDTH, SOURCE_HEIGHT, umbColumnaR, Labeling);
    }

    //si descomentas las funciones de labeling da el error
 }


 uint8_t RGB2HSV(uint8_t r, uint8_t g, uint8_t b) {
   unsigned char min, max, delta;
```

```
  uint8_t h;

  if (r < g)min = r; else min = g;
  if (b < min)min = b;

  if (r > g)max = r; else max = g;
  if (b > max)max = b;


  delta = max - min;

  if ( max == 0 ) {
    h = 0;
    return h;
  }

 if(delta==0) delta=1;



  if ( r == max )
    h = (g - b) * 60 / delta;    // between yellow & magenta
  else if ( g == max )
    h = 120 + (b - r) * 60 / delta; // between cyan & yellow
  else
    h = 240 + (r - g) * 60 / delta; // between magenta & cyan

  if ( h < 0 )
    h += 360;

  return h;

 }

 void LabelImage(uint32_t width, uint32_t height, uint8_t** input, uint8_t** output)
{
uint32_t labelNo = 0;
 for (uint32_t y = 0; y < height-1; y++)
 {
  for (uint32_t x = 0; x < width-1; x++)
  {
   if (input [y][x] == 0) continue;   /* This pixel is not part of a component */
   if (output[y][x] != 0) continue;   /* This pixel has already been labelled  */
   /* New component found */
   labelNo++;
   LabelComponent(width, height, input, output, labelNo, x, y);
  }
 }
}

void LabelComponent(uint32_t width, uint32_t height, uint8_t**  input, uint8_t**  output, uint8_t labelNo, uint32_t x, uint32_t
y)
{
```

```
  if (input [y][x] == 0) return;   /* This pixel is not part of a component */
  if (output[y][x] != 0) return;   /* This pixel has already been labelled  */
  output[y][x] = labelNo;

  /* Now label the 4 neighbours: */
  if (x > 0)        LabelComponent(width, height, input, output, labelNo, x-1, y);   /* left  pixel */
  if (x < width-1)  LabelComponent(width, height, input, output, labelNo, x+1, y);   /* right pixel */
  if (y > 0)        LabelComponent(width, height, input, output, labelNo, x, y-1);   /* upper pixel */
  if (y < height-1) LabelComponent(width, height, input, output, labelNo, x, y+1);   /* lower pixel */
}
```

UNIVERSITAT POLITÈCNICA DE VALÈNCIA

Escuela Técnica Superior de Ingeniería del Diseño

ANEXO A-6: Código Arduino para el cálculo del centro de gravedad de la imagen embebida en la ESP32-CAM

DISEÑO Y PROGRAMACIÓN DE UN SISTEMA DE BAJO COSTE BASADO EN LA ESP 32-CAM PARA EL CONTROL DE UN BRAZO ROBÓTICO

TRABAJO FINAL DEL

Grado en Ingeniería Electrónica Industrial y Automática

REALIZADO POR

Juan José Martín Osuna

TUTORIZADO POR

Leopoldo Armesto Ángel

Ángel Rodas Jordá

CURSO ACADÉMICO: 2021/2022

```
// Change according to your model
// The models available are
//   - CAMERA_MODEL_WROVER_KIT
//   - CAMERA_MODEL_ESP_EYE
//   - CAMERA_MODEL_M5STACK_PSRAM
//   - CAMERA_MODEL_M5STACK_WIDE
//   - CAMERA_MODEL_AI_THINKER
#define CAMERA_MODEL_AI_THINKER

#include <FS.h>
#include <SPIFFS.h>
#include "EloquentVision.h"


#define FRAME_SIZE FRAMESIZE_QVGA
#define SOURCE_WIDTH 320
#define SOURCE_HEIGHT 240

#define led_FLASH 4

using namespace Eloquent::Vision;
using namespace Eloquent::Vision::IO;
using namespace Eloquent::Vision::ImageProcessing;

// an easy interface to capture images from the camera
ESP32Camera camera;


uint8_t** umbColumnaR = (uint8_t**)malloc(SOURCE_HEIGHT * sizeof(uint8_t*));
uint8_t** umbColumnaA = (uint8_t**)malloc(SOURCE_HEIGHT * sizeof(uint8_t*));

//Funcion auxiliar
uint8_t h,  centroidex, centroidey, Yreal, Xreal;

uint32_t auxiliarx=0, auxiliary=0,contadorR = 0, contadorA = 0;

void setup() {
  Serial.begin(115200);
  camera.begin(FRAME_SIZE, PIXFORMAT_RGB565);

  pinMode(led_FLASH, OUTPUT);

  for ( uint32_t i = 0; i < SOURCE_HEIGHT ; i++) {

     umbColumnaR[i]=(uint8_t*)malloc(SOURCE_WIDTH * sizeof(uint8_t));
     umbColumnaA[i]=(uint8_t*)malloc(SOURCE_WIDTH * sizeof(uint8_t));
  }

   for (unsigned short y = 0; y < SOURCE_HEIGHT; y++)
    for (unsigned short x = 0; x < SOURCE_WIDTH; x++){
       umbColumnaR[y][x] = 0;
       umbColumnaA[y][x] = 0;
```

```
  }
}

 /**

 */
 void loop() {
  digitalWrite(led_FLASH, HIGH);
  uint32_t start = millis();
  camera_fb_t *frame = camera.capture();

   uint32_t desplazarX = 0;
   uint32_t desplazarY = 0;
    desplazarX = 0;
   desplazarY = 0;
      contadorR = 0;
      contadorA = 0;



  Serial.print("Foto tomada ");
  for ( uint32_t e = 0; e < (SOURCE_WIDTH * SOURCE_HEIGHT * 2)-2; e = e + 2) {



    uint8_t high = frame->buf[e];
    uint8_t low  = frame->buf[e + 1];
    uint16_t pixel = (high << 8) | low;

    uint8_t r = (pixel & 0b1111100000000000) >> 11;
    uint8_t g = (pixel & 0b0000011111100000) >> 5;
    uint8_t b = (pixel & 0b0000000000011111);

    //Serial.print("Red: ");
   // Serial.print(r);
    //Serial.print(", Green: ");
    //Serial.print(g);
    //Serial.print(", Blue: ");
    //Serial.println(b);

    if(r==0) r=0.000001;
    if(g==0) r=0.000001;
    if(b==0) r=0.000001;

    // redefinición rango de colores
    uint8_t R = (255 * r) / 31;
    uint8_t G = (255 * g) / 63;
    uint8_t B = (255 * b) / 31;
    //pasar a h de hasv
    h = RGB2HSV(R, G, B);
```

```
    if ((h > 0 && h < 50) || (h > 170 && h < 180))
    {
      //Serial.print("pixel rojo");
      contadorR++;
     umbColumnaR[desplazarY] [desplazarX] = 255;
     umbColumnaA[desplazarY] [desplazarX] = 0;
    }
    else if (h > 230 && h < 255) {
      //Serial.print("pixel azul");
      contadorA++;
     umbColumnaR[desplazarY] [desplazarX] = 0;
     umbColumnaA[desplazarY] [desplazarX] = 255;
    }
    else
    {
     umbColumnaR[desplazarY] [desplazarX] = 0;
     umbColumnaA[desplazarY] [desplazarX] = 0;

    }


  desplazarX++;

 if( desplazarX == (SOURCE_WIDTH - 1))
 {

  desplazarX = 0;
  desplazarY++;
 }

  }
Serial.print("final imagen ");

 //algoritmo centro de gravedad

 if(contadorR>contadorA){
   Serial.print("pieza roja");

 for (unsigned short y = 0; y < SOURCE_HEIGHT-1; y++)
 {
   for (unsigned short x = 0; x < SOURCE_WIDTH-1; x++)
   {

    if(umbColumnaR[y] [x]==255){
     auxiliarx = auxiliarx + x;
     auxiliary = auxiliarx + y;
     centroidex = auxiliarx/contadorR;
     centroidey = auxiliary/contadorR;
    }
 }
 }
 }
 else{
```

```
  Serial.print("pieza azul");
   for (unsigned short y = 0; y < SOURCE_HEIGHT-1; y++)
{
  for (unsigned short x = 0; x < SOURCE_WIDTH-1; x++)
  {

   if(umbColumnaA[y] [x]==255){
    auxiliarx = auxiliarx + x;
    auxiliary = auxiliarx + y;
    centroidex = auxiliarx/contadorA;
    centroidey = auxiliary/contadorA;
}
  }
}
}
 //tasformación a dimensiones reales
 //alto total 150mm. ancho total 170 mm
Yreal= (centroidey * 150)/SOURCE_HEIGHT;
Xreal=  (centroidex* 150)/SOURCE_WIDTH;
// Serial.println(Yreal);
  // Serial.println(Xreal);
Serial.println("fin del procesado");
  delay(1000);

}



uint8_t RGB2HSV(uint8_t r, uint8_t g, uint8_t b) {
  unsigned char min, max, delta;
  uint8_t h;

  if (r < g)min = r; else min = g;
  if (b < min)min = b;

  if (r > g)max = r; else max = g;
  if (b > max)max = b;


  delta = max - min;

  if ( max == 0 ) {
    h = 0;
    return h;
  }

 if(delta==0) delta=1;



  if ( r == max )
    h = (g - b) * 60 / delta;    // between yellow & magenta
  else if ( g == max )
```

```
    h = 120 + (b - r) * 60 / delta; // between cyan & yellow
  else
    h = 240 + (r - g) * 60 / delta; // between magenta & cyan

  if ( h < 0 )
    h += 360;

  return h;

}
```

UNIVERSITAT POLITÈCNICA DE VALÈNCIA

Escuela Técnica Superior de Ingeniería del Diseño

ANEXO A-7: Programación de la ESP 32-CAM mediante el IDE de arduino

DISEÑO Y PROGRAMACIÓN DE UN SISTEMA DE BAJO COSTE BASADO EN LA ESP 32-CAM PARA EL CONTROL DE UN BRAZO ROBÓTICO

TRABAJO FINAL DEL

Grado en Ingeniería Electrónica Industrial y Automática

REALIZADO POR

Juan José Martín Osuna

TUTORIZADO POR

Leopoldo Armesto Ángel

Ángel Rodas Jordá

CURSO ACADÉMICO: 2021/2022

La ESP 32 -CAM es un periférico que se puede programar de muchas manera, pero para este proyecto se hará desde el IDE de arduino, los motivos son principalmente el hecho de que ya se ha trabajado con este programa para etapas anteriores del proyecto lo cual implica que no es necesario contar con más software adicional y por otro el hecho de que en las propias herramientas de desarrollo que se deberán instalar para poder trabajar con la cámara, ya se cuenta con un programa de servidor web como el que se necesita, haciendo que no sea necesario programarlo.

Es importante saber que para la programación de esta cámara es necesario contar con un programador que comunique y conecte la cámara y el ordenador. Existen muchos modelos compatibles con este periférico, como por ejemplo, el "esp32 cam MB" (figura 39) o el "ft232rl" (figura 40):

Figura 39. Foto del programador esp32 cam MB

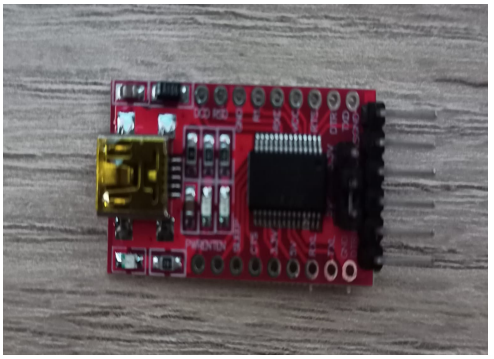


Figura 40. Foto del programador ft232rl

En este caso, se ha utilizado el modelo "esp32 cam MB", ya que, se integra directamente sobre la cámara y facilita las programación de la misma sin necesidad de puentes o conexiones adicionales.

Una vez definido el motivo de selección del entorno de programación se describirán los pasos a seguir.

El primer paso consistirá en instalar al programa las herramientas propias de los chips ESP 32. Para ello, se abrirá la pestaña superior de "Archivo" y se seleccionará "Preferencias". En la pestaña emergente que aparecerá, se clica el botón perteneciente al apartado de "Gestor de URLs Adicionales de tarjetas".

Tras esto, aparecera otro menu en el que se deberá copiar la siguiente URL:

https://dl.espressif.com/dl/package_esp32_index.json

Es importante que, si dentro de este menú se encuentran otras direcciones diferentes a esta, no se han de borrar, ya que aunque no interfieran con el funcionamiento de la cámara, sí lo pueden hacer con otros programas ya creados, generando incompatibilidades.

Una vez copiado, se presiona "ok" en ambos menús. Con esto, el programa ya podrá buscar las tarjetas e instalar las bibliotecas. Para ello, se seleccionará el menú herramientas. Dentro del mismo se presionara placa y después, gestor de tarjetas.

En la barra de búsqueda del menú que aparece, se ha de buscar "esp 32" e instalar el pack de tarjetas de la familia esp 32.

Una vez hecho esto, dentro de nuevo del menú herramientas y el sub menu placa, aparecerá un nuevo sub menu, el " esp32 Arduino". Dentro de ese menú se seleccionara la placa "ai tinker esp32 cam". Se modificarán el resto de parámetros de subida de la placa, hasta que sean como los que aparecen en la imagen (figura 41).

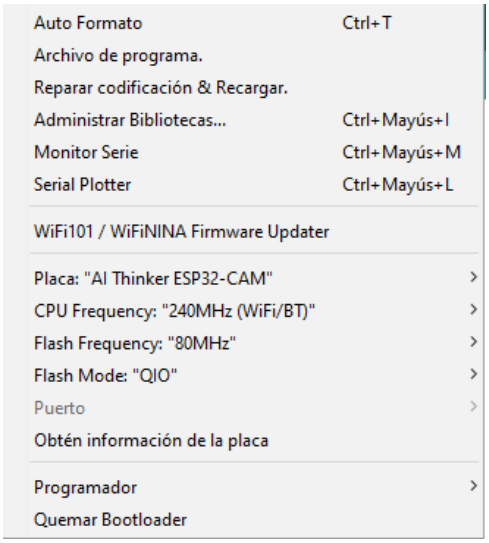


Figura 41. Imagen de la configuración del IDE de Arduino para subir los programas a la ESP 32 - CAM

En este punto, se concretará la cámara y el programador al ordenador. Una vez hecho esto, de nuevo en el menú herramientas, deberá aparecer el submenú puerto en otro color y se tendrá que seleccionar aquel en el que se ha conectado la cámara.

Se puede dar el caso de que el ordenador no detecte la cámara. Si no lo hace, es importante comprobar que se está usando un cable que permita trasmitir información, no solo carga. Si el cable es el correcto, el problema puede venir de no disponer de los drivers necesarios para comunicar el puerto USB con el programador. Para ello, comprobaremos el modelo de chip de conexión que tiene el programador e instalaremos los drivers para ese modelo. Para este programador el chip es el “ch340g”.

Tras instalar estos drivers, el ordenador detectará la cámara y se podrá seleccionar el puerto, en el menú herramientas, como se ha explicado con anterioridad.

Una vez realizado todo este proceso, ya se puede proceder a subir un programa a la cámara, presionando el botón subir del IDE de Arduino.

Para esta etapa del proyecto caso se subirá el ejemplo de la cámara para que actúe como servidor web, con el que cuenta el paquete esp32. Se hacedera a éste desde el menú “archivos”, submenú “ejemplos”, y se seleccionara aquellos para esp32, dentro de la variedad de ejemplos hay una sección para cámara y dentro el ejemplo “web server”.

Antes de poder subir este código a la placa, es necesario realizar ciertas modificaciones.

Por un lado en la parte superior del código, aparece la sección de modelo de cámara, por defecto viene seleccionado el más genérico, en este caso y siguiendo con la selección que se ha hecho por lo que respecta a la cámara, se comentará esta línea y se descomenta la línea para el modelo “CAMERA_MODEL_AI_THINKER”.

Después de esto, se deberá definir en las variables ssid y password, el nombre y la contraseña de la red wifi a la que se conectará la cámara. Es muy importante asegurarse que el nombre de la red wifi y la contraseña estén escritos igual, así como que la banda AP de la red sea de 2.4 GHz. Para poder recibir la imagen desde el ordenador, la cámara ha de estar conectada a la misma red wifi, ya que de lo contrario no se podrá acceder. Tras esto, ya se podrá proceder a subir el programa a la cámara.

Es importante durante el proceso de subida estar atentos, ya que en cierto momento, aparecerá una línea punteada en el monitor del programa. En ese momento, será necesario pulsar el botón de reset (figura 42),que se encuentra en la parte trasera de la cámara. Tras esto la línea se detendrá y dará paso a la subida.

Figura 42. Imagen para la identificación del botón de reset en la ESP 32 - CAM

Tras un tiempo, aparecerá el mensaje de “subido” en el monitor de Arduino. Con esto se confirma que se ha realizado la subida del programa de forma correcta a la cámara.

A continuación se abrirá el monitor serie, y se volverá a presionar el botón reset. Con esto aparecerá información de la cámara por el monitor serie y comenzará el proceso de conexión a la red wifi que se le ha configurado anteriormente.

Una vez esté conectado, en el monitor serie aparecerá el mensaje de conectado, así como, el puerto que está habilitado para streaming y la dirección a la que acceder.

Por último, y para comprobar que la cámara está funcionando correctamente, se buscará la dirección http, que aparece en el monitor serie en el buscador, si el proceso se ha realizado correctamente y tanto la cámara como el ordenador están conectadas a la misma red wifi, se podrá ver la imagen que envía la cámara.

UNIVERSITAT POLITÈCNICA DE VALÈNCIA

Escuela Técnica Superior de Ingeniería del Diseño

PLANOS

DISEÑO Y PROGRAMACIÓN DE UN SISTEMA DE BAJO COSTE BASADO EN LA ESP 32-CAM PARA EL CONTROL DE UN BRAZO ROBÓTICO

TRABAJO FINAL DEL

Grado en Ingeniería Electrónica Industrial y Automática

REALIZADO POR

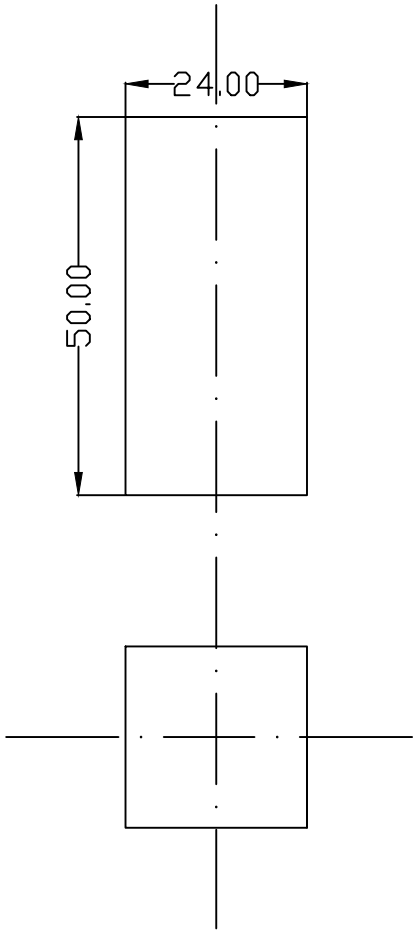
Juan José Martín Osuna

TUTORIZADO POR

Leopoldo Armesto Ángel

Ángel Rodas Jordá

CURSO ACADÉMICO: 2021/2022



PROYECTO: DISEÑO Y PROGRAMACIÓN DE UN SISTEMA DE BAJO COSTE BASADO EN LA ESP 32-CAM PARA EL CONTROL DE UN BRAZO ROBÓTICO

TITULAR: Martín Osuna, Juan José

Referencia: 070522-1

Fecha: 07/05/22

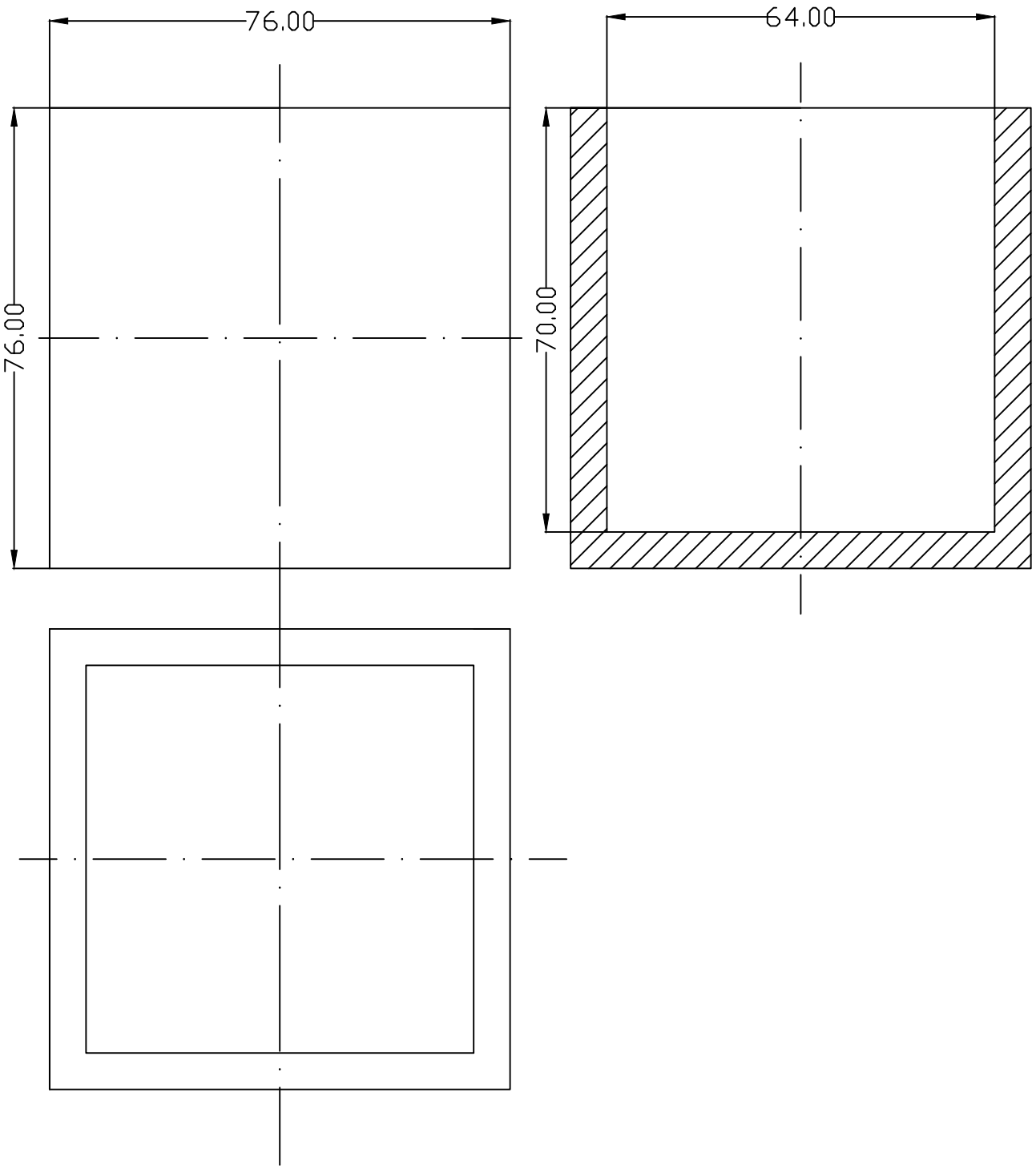
Escala: 1:1

Autor: Martín Osuna, Juan José

Titulación: Ing. electrónica industrial y automática

Plano: Pieza de color

Plano Nº: 01



PROYECTO: DISEÑO Y PROGRAMACIÓN DE UN SISTEMA DE BAJO COSTE BASADO EN LA ESP 32-CAM PARA EL CONTROL DE UN BRAZO ROBÓTICO

TITULAR: Martín Osuna, Juan José

Referencia: 070522-2

Fecha: 07/05/22

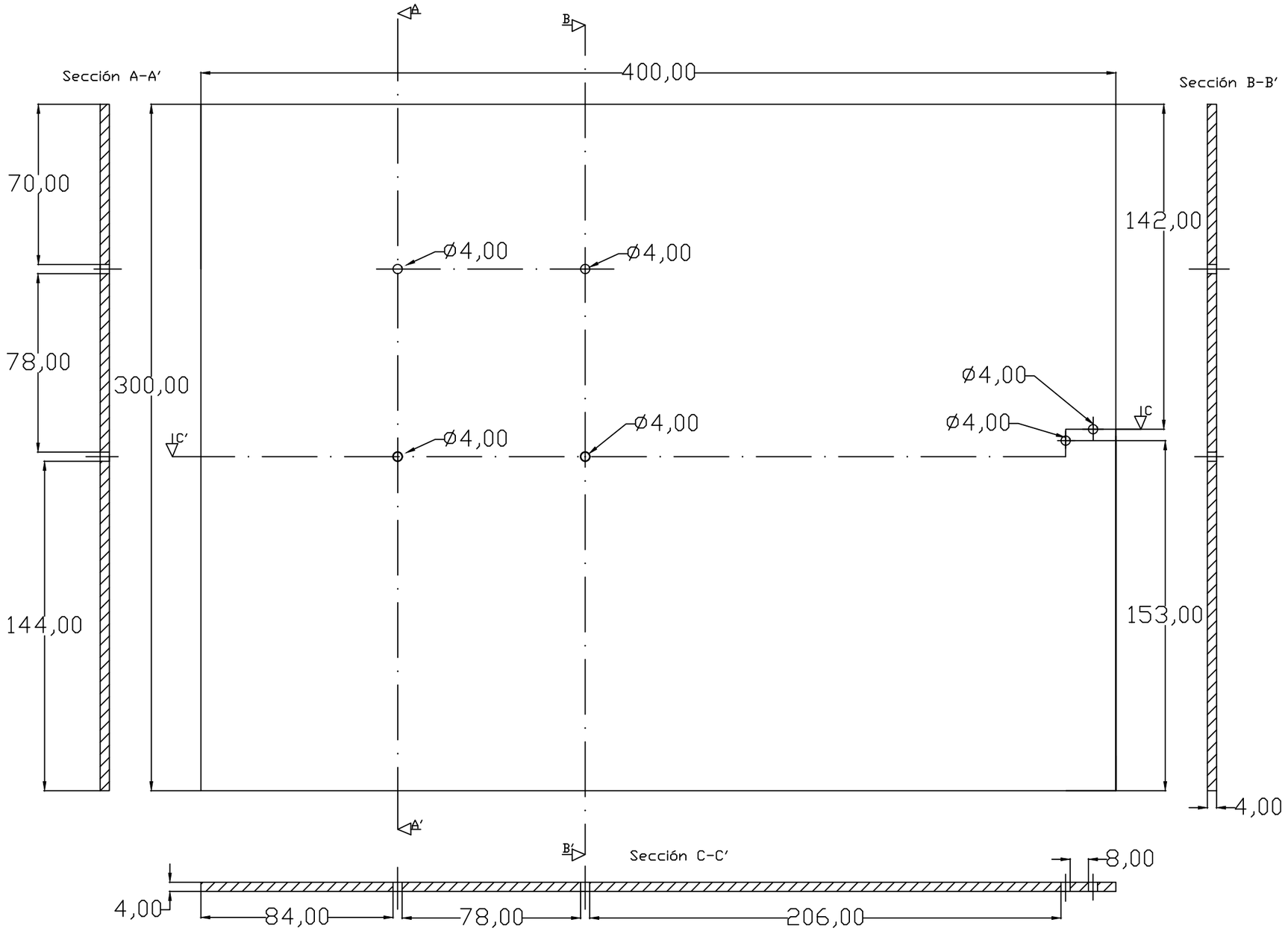
Escala: 1:1

Autor: Martín Osuna, Juan José

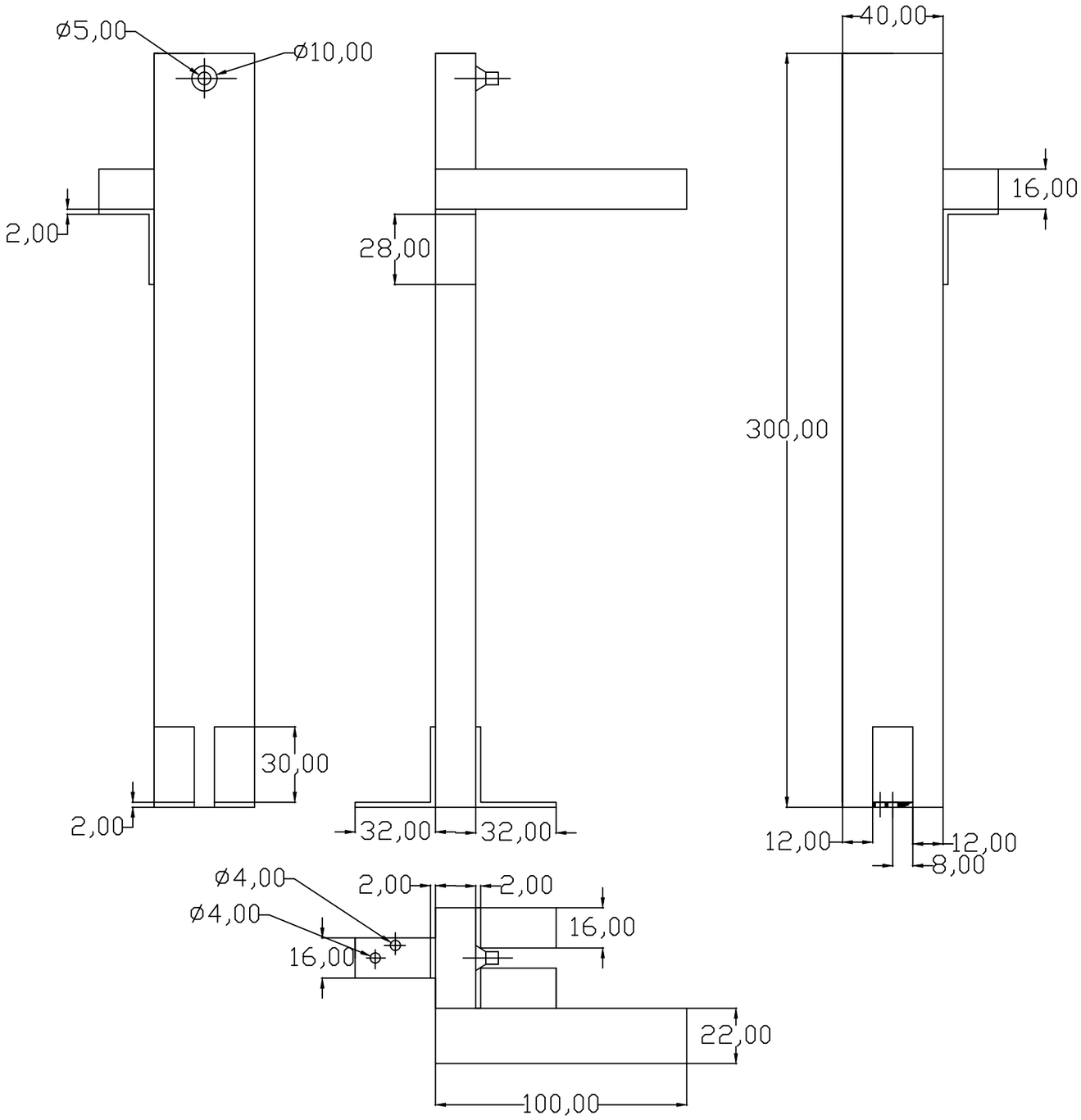
Titulación: Ing. electrónica industrial y automática

Plano: Recipientes

Plano Nº: 02



PROYECTO: DISEÑO Y PROGRAMACIÓN DE UN SISTEMA DE BAJO COSTE BASADO EN LA ESP 32-CAM PARA EL CONTROL DE UN BRAZO ROBÓTICO TITULAR: Martín Osuna, Juan José Referencia: 070522-3	Fecha:07/05/22 Escala: 1:2
Autor: Martín Osuna, Juan José Titulación: Ing. electrónica industrial y automática	Plano: Base de fijación Plano Nº: 03



PROYECTO: DISEÑO Y PROGRAMACIÓN DE UN SISTEMA DE BAJO COSTE BASADO EN LA ESP 32-CAM PARA EL CONTROL DE UN BRAZO ROBÓTICO

TITULAR: Martín Osuna, Juan José

Referencia: 070522-4

Fecha: 07/05/22

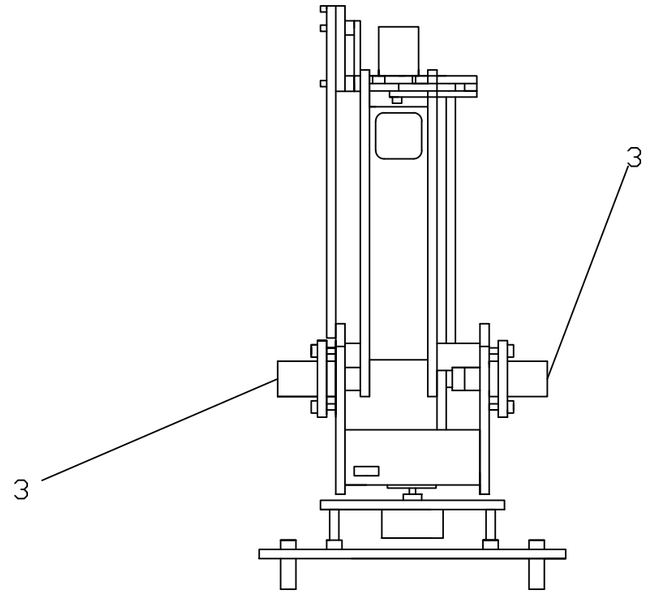
Escala: 1:2

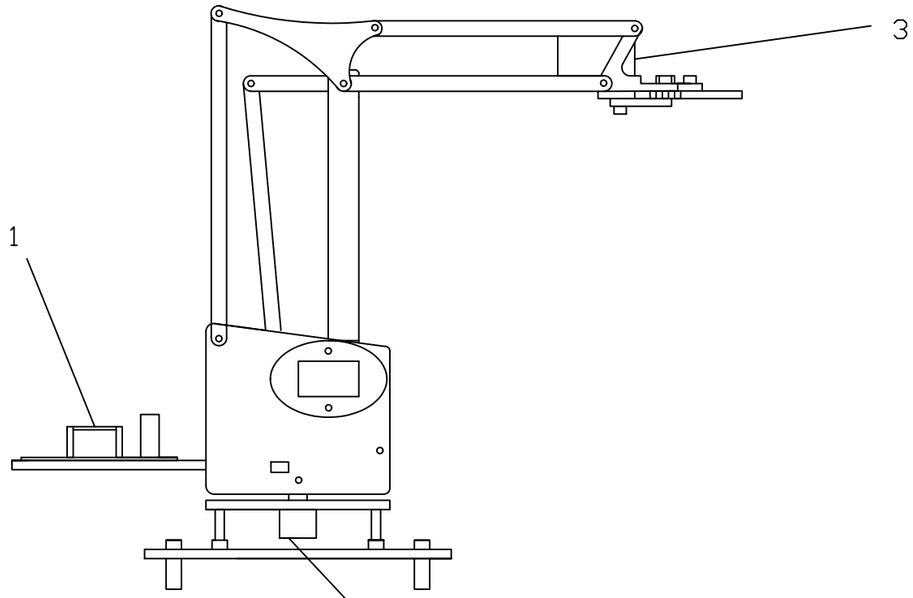
Autor: Martín Osuna, Juan José

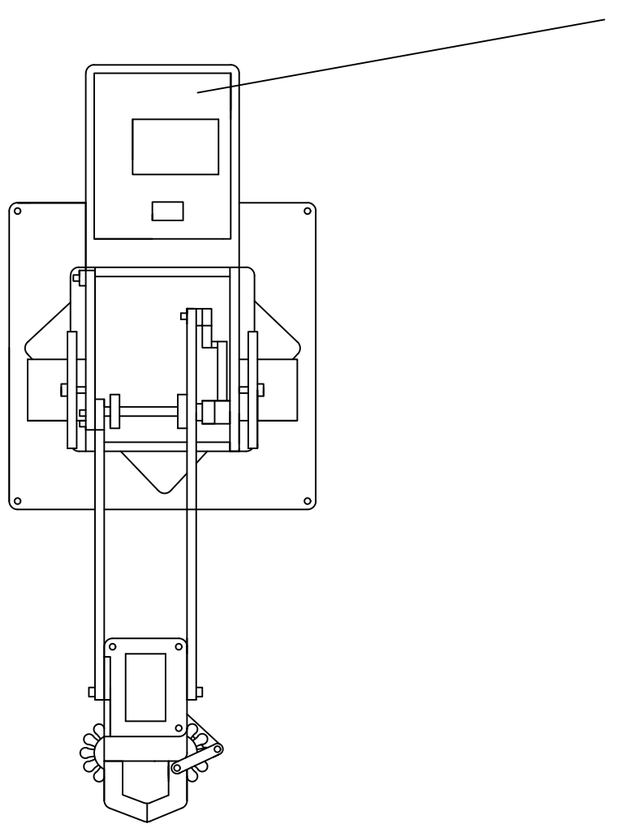
Titulación: Ing. electrónica industrial y automática

Plano: Soporte para cámara

Plano Nº: 04







Nº	CANT.	NOMBRE DEL OBJETO	REFERENCIA	MATERIAL
3	4	Servo	MG90S	–
2	1	Extensión shield	Arduino Nano	–
1	1	Arduino Nano	v3.0	–

PROYECTO: DISEÑO Y PROGRAMACIÓN DE UN SISTEMA DE BAJO COSTE BASADO EN LA ESP 32-CAM PARA EL CONTROL DE UN BRAZO ROBÓTICO

TITULAR: Martín Osuna, Juan José

Referencia: 070522-5

Fecha:04/06/22

Escala: 1:2

Autor: Martín Osuna, Juan José

Titulación: Ing. electrónica industrial y automática

Plano: Robot paralelogramo

Plano Nº: 05

UNIVERSITAT POLITÈCNICA DE VALÈNCIA

Escuela Técnica Superior de Ingeniería del Diseño

PLIEGO DE CONDICIONES

DISEÑO Y PROGRAMACIÓN DE UN SISTEMA DE BAJO COSTE BASADO EN LA ESP 32-CAM PARA EL CONTROL DE UN BRAZO ROBÓTICO

TRABAJO FINAL DEL

Grado en Ingeniería Electrónica Industrial y Automática

REALIZADO POR

Juan José Martín Osuna

TUTORIZADO POR

Leopoldo Armesto Ángel

Ángel Rodas Jordá

CURSO ACADÉMICO: 2021/2022

3.1. Objeto

Se procederá a la descripción de los materiales necesarios para el montaje y la reproducción del sistema descrito en este proyecto. Así como, especificar los pasos a seguir para su montaje y las comprobaciones posteriores.

La detección de los objetos se hará en primera instancia mediante la webcam modelo hama c-400 para más adelante ser sustituida por la Esp 32-CAM.

Se usará el entorno de arduino en todo momento para el control del robot y adicionalmente en las últimas etapas para la programación de la cámara. Por su parte todos los algoritmos de visión artificial no embebido en la cámara se harán empleando el lenguaje de programación Python, con su biblioteca OpenCV.

El proyecto no precisa de mantenimiento y en caso de rotura o desgaste de algún componente se facilitará su referencia de compra o sus planos para su sustitución.

3.2. Normativa

La normativa que se ha tenido en cuenta para la redacción del proyecto es la siguiente:

- **ISO/IEC 80000**, normativa que regula el uso de las magnitudes físicas y las unidades de medida. A la hora de representar estos elementos se ha hecho siguiendo esta guía de estilo.

- **UNE 157001:2014**, normativa que regula los criterios generales para la redacción de un proyecto técnico. La redacción y estructura de este documento se ha realizado siguiendo la normativa vigente, de esta norma.

- **UNE-EN 310:1994**, normativa que determina la resistencia en flexión y la elasticidad de los tableros derivados de la madera. Usado para determinar la viabilidad del tablero de madera usado como base del proyecto.

- **Real Decreto 186/2016, de 6 de mayo**, por el que se regula la compatibilidad electromagnética de los equipos eléctricos y electrónicos. Muy relevante en este proyecto por el trabajo simultáneo de varios procesadores y comunicación wifi simultánea, y en espacios cercanos.

3.3. Materiales

A continuación se listarán los componentes principales a utilizar, así como las características que han de cumplir a la hora de ser utilizados para replicar el proyecto.

3.3.1. Características principales

- **Materiales eléctricos:**

    - Módulo de cámara ESP 32-cam, con procesador Xtensa Dual-Core 32-bit y 520 KB de memoria SRAM. A debe ser compatible con conexión wifi para permitir la comunicación con el ordenador.

- Placa arduino nano 3.0, basado en el chip ATmega328, que cuente con memoria Flash de 32KB, una SRAM de 2 KB y una EEPROM de 1KB. Es necesario que disponga de todos los pines soldados para ser compatibles con cables tipo Dupont.

- Modelo de webcam Hama c400, con resolución 1920x1080 y cable con conexión USB-A .

- 4 servomotores modelo MG90S, con una tensión de trabajo de 3 a 7.2 voltios, dimensiones de 22 x 11,5 x 27 mm, engranajes metálicos y torque en reposo de 2,2 kg x cm.

- Shield para placa de expansión nano 3.0, ha de contar con tomas por pin, ya que no solo ha de transmitir información sino tensión a los servos. A su vez, ha de contar con un pin de alimentación de 5V, para alimentar la cámara en caso de ser necesario, así como entrada de jack para alimentación.

- **Materiales eléctricos:**

  - La conexión entre módulos se hará mediante cables debles Dupont, ya que son los más utilizados para arduino y son compatibles con el shield para los servos.

  - Cable USB 2.0 de tipo A a tipo B mini, este cable será el utilizado para comunicar el arduino con el ordenador, tanto para trabajar con el puerto serie como para cargar el programa.

  - Cable USB 2.0 de tipo A A a tipo A micro, este cable será el utilizado para comunicar la ESP32-CAM con el ordenador, principalmente, para cargar el programa y alimentarla.

- **Elementos de poliestireno expandido:**

  - Debe de disponerse de material suficiente para fabricar 4 de las piezas definidas en el plano 2.1. de apartado de planos. Su densidad no ha de superar los 1050 kg / m^3.

- **Elementos de madera:**

  - El espesor del contrachapado no debe superar en ningún caso los 4 mm y tiene que tener una densidad mínima de 500 km / m^3.

  - El listón de madera maciza ha de tener ha de disponer de una longitud suficiente para poder construir la totalidad del soporte definido en el plano 2.4. del apartado de planos. A su vez, ha de ser una madera lo suficientemente blanda para poder ser perforada con un tornillo para madera sin necesidad de utilizar herramientas eléctricas.

3.3.2. Control de calidad

A continuación se detallan los requisitos mínimos de calidad que deben cumplir todos los componentes del proyecto.

Por los que respetan a los las cámaras y la placa de arduino, se ha de comprobar su funcionamiento antes del montaje, alimentándose de la forma que se indique para cada dispositivo. A su vez, es necesario que no presenten signos de quemadura que se pueden haber generado como resultado de un cortocircuito.

Se comprobarán los cables de conexión del arduino para asegurarse que son cable de alimentación y datos, ya que de lo contrario no se podrán cargar los programas. A su vez, se ha de comprobar que no presentan ningún tipo de rotura que pueda producir un cortocircuito más adelante.

Por parte de la madera, se ha de asegurar que no presenta ningún tipo de agujero o deterioro, ya sea natural o provocado en su fabricación.

3.4. Condiciones de ejecución

En este apartado se describe el proceso de ejecución que se ha de realizar para replicar el proyecto. Así como los controles de calidad pertinentes que se han de hacer una vez montado.

1. El primer paso será hacer los agujeros necesarios en la plancha de contrachapado, que se utilizará como base. Estos agujeros se harán empleando un taladro eléctrico, con una broca para madera de métrica 3, la posición de los agujeros vendrá definida en el plano 2.3. del apartado de planos.

2. Se pintará la plancha de contrachapado con pintura blanca, con el objetivo de que a la hora de tomar la imagen, el color de la madera no interfiera.

3. Tras esto se fijará el robot en los agujeros que forman un cuadrado, utilizando tornillos de métrica tres sin punta.

4. A continuación se construirá el soporte para las cámaras, para esto se cortaran dos piezas del listón de madera, siguiendo las dimensiones indicadas en el plano 2.4. del apartado de planos. Tras esto se situarán las escuadras en los puntos definidos en ese mismo plano y se fijarán al cuerpo de madera mediante tornillos de métrica 3, con punta de madera. Tras esto se situará el tornillo para la webcam en la posición designada en el plano con ayuda de silicona termofusible.

5. Se fijará el soporte a la base de contrachapado mediante dos tornillos de métrica 3 y sus respectivas tuercas.

6. Después se cortarán dos piezas de poliestireno expandido, hasta darle las dimensiones indicadas en el plano 2.1. del apartado de planos. para posteriormente pintar una de las caras cuadradas de la pieza con pintura azul y otra con pintura roja.

7. Se pintarán los recipientes definidos en el plano 2.2 del apartado de planos, uno de rojo y otro de azul. Tras su secado se situarán en el lado derecho a 80 mm del robot.

8. El siguiente paso será colocar la ESP 32-CAM en el mástil saliente del soporte para cámaras, es importante colocarla de forma que la lente apunte a la base y la clavija de conexión del cable apunte hacia fuera.

9. Llegados a este punto la distribución de los elementos ya está completa y ha de ser igual a la mostrada en la siguiente imagen (figura 43).

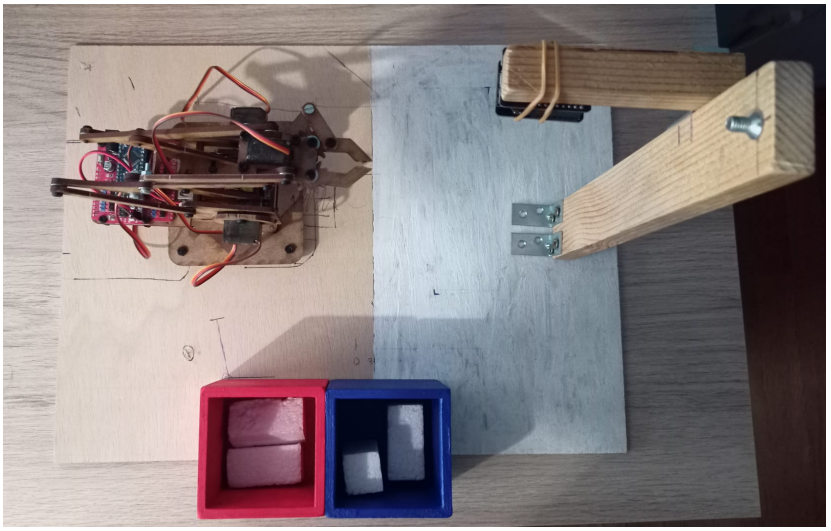


Figura 43. Imagen cenital tomada para referenciar la posición final de los elementos que constituyen el proyecto

   Si algún elemento no cuadra en posición o forma con lo mostrado en esta imagen será necesario repetir los pasos de ejecución propios de dicho elemento para asegurar la correcta distribución de los elementos antes de cargar los programas.

10. Se conectará la placa de arduino, mediante el cable USB 2.0 de tipo A a tipo B mini, al ordenador. Tras comprobar que se enciende la placa arduino, se cargará el programa descrito en el anexo A-3 desde el IDE de arduino.

11. Se conectará la cámara ESP32-CAM, mediante el cable Cable USB 2.0 de tipo A A a tipo A micro. A continuación se cargará el programa de ejemplo de servidor web, propio de la librería de ESP-32, desde el IDE de arduino.

12. Tras esto se abrirá y ejecutará en el mismo ordenador al que están conectados la placa de arduino y la cámara ESP32-CAM, el código de python descrito en el apartado de anexos A-4.

3.5. Control de calidad

Antes de comprobar el funcionamiento del proyecto es necesario asegurarse de que todos los elementos estén conectados correctamente, así como que la ESP32-CAM y el ordenador al que esté conectada estén, a su vez, conectados a la misma red wifi.

Es importante comprobar que la dirección de conexión que proporciona la cámara al cargarle el programa y la dirección de conexión descrita en el código de python son la misma, ya que de lo contrario, el programa de python dará un error a la hora de conectarse haciendo imposible el funcionamiento.

Tras estas comprobaciones, el sistema se encuentra disponible para realizar su tarea, por ello se realizará la siguientes pruebas, se probara que el robot hace la tarea correcta de almacenaje, solo con la pieza azul 10 veces y posteriormente lo mismo solo con la pieza roja.

Por último y para asegurarse de que el criterio de preferencia de orden se cumple se situarán ambas piezas en escena, colocando cinco veces la pieza azul por delante de la roja y otra cinco la pieza roja por delante de la azul. Estas pruebas permiten determinar no solo el correcto funcionamiento del sistema sino también posibles problemas de calentamiento o descalibrado en la cámara.

UNIVERSITAT POLITÈCNICA DE VALÈNCIA

Escuela Técnica Superior de Ingeniería del Diseño

PRESUPUESTO

DISEÑO Y PROGRAMACIÓN DE UN SISTEMA DE BAJO COSTE BASADO EN LA ESP 32-CAM PARA EL CONTROL DE UN BRAZO ROBÓTICO

TRABAJO FINAL DEL

Grado en Ingeniería Electrónica Industrial y Automática

REALIZADO POR

Juan José Martín Osuna

TUTORIZADO POR

Leopoldo Armesto Ángel

Ángel Rodas Jordá

CURSO ACADÉMICO: 2021/2022

4.1. Tablas de costes por naturaleza del elemento

En las siguientes tablas se desglosarán el coste de los distintos elementos que constituyen el proyecto según su naturaleza.

ROBOT					
MATERIALES					
Uds	Denominación	Cantidad	Precio(€)		Total
Electrónica					
u	Arduino Nano v3.0.	1	4,5		4,5
u	I/O Extension Shield para Arduino Nano	1	4,5		4,5
u	Batería 3.7V 16340 Li-Ion 2800mAh	1	2		2
u	Cable Dupont 10 cm Macho-Hembra	5	0,1		0,5
u	Soporte Batería 16340 Simple Powerbank	1	5		5
Motricidad					
u	Servo MG90S	4	4		16
Tornilleria					
u	Tornillo rosca-chapa 2x6mm	10	0,05		0,5
u	Tornillo M3x10mm (Plástico Negro)	10	0,05		0,5
u	Tornillo M3x6mm (Plástico Negro)	1	0,05		0,05
u	Tornillo M3x8mm (Plástico Negro)	30	0,05		1,5
u	Arandela M3 Acero Inoxidable	20	0,05		1
u	Tuerca Hexagonal M3 (Plástico Negro)	20	0,05		1
u	Tornillo M3x16mm Acero Inoxidable	4	0,1		0,4
u	Separador M3x10 Plásico Negro	4	0,1		0,4
Estructura					
u	kit de Piezas de Corte Láser para robot meArm	1	10		10
SUBTOTAL MATERIALES					47,85

CÁMARAS					
MATERIALES					
Uds	Denominación	Cantidad	Precio(€)		Total
Dispositivos					
u	ESP 32 - CAM	1	4,72		4,72
u	Webcam hama c-400	1	34,99		34,99
Componentes para la programación					
u	Programador esp32 cam MB	1	1,06		1,06
Cables de conexión					
u	Cable USB 2.0 de tipo A a tipo B mini	1	5,99		5,99
u	Cable USB 2.0 de tipo A a tipo A micro	1	6,99		6,99
SUBTOTAL MATERIALES					49,03

SOPORTE CAMARA					
MATERIALES					
Uds	Denominación	Cantidad	Precio(€)		Total
Materia prima					
m	Listón de madera de 300 x 40 mm	1	2,6		2,6
Tornilleria					
u	Tornillo rosca-chapa 2x10mm	8	0,1		0,8
u	Tornillo M3x20mm Acero Inoxidable	2	0,1		0,2
u	Tuerca Hexagonal M3 (Acero Inoxidable)	2	0,1		0,2
u	Tornillo de 1/4, para tripode de camara	1	0,82		0,82
u	Escuadra en L de 25 mm	4	0,25		1
SUBTOTAL MATERIALES					5,62

OTROS

MATERIALES					
Uds	Denominación	Cantidad	Precio(€)		Total
Materia prima					
u	Lámina de contrachapado 300 x 400 mm	1	1,25		1,25
u	Recipientes de madera	2	1,5		3
u	Cubo poliestireno expandido 45 x 45 mm	1	0,75		0,75
SUBTOTAL MATERIALES					5

MANO DE OBRA

MATERIALES					
Uds	Denominación	Cantidad	Precio(€/h)		Total
h	Ingeniero electronico	50	15		750
h	Ingeniero informático	200	20		4000
h	Técnico de montaje	20	8		160
SUBTOTAL MATERIALES					4910

COSTES INDIRECTOS

MATERIALES					
Uds	Denominación	Cantidad (%)	Precio(€)		Total
u	Medios auxiliares sobre coste: ordenador, herramientas, software, estaño,pintura,...	15	5017,5		752,63
SUBTOTAL MATERIALES					752,63

4.2. Tablas resumen

A continuación se muestra la tabla resumen con el coste total del proyecto, desglosado por la naturaleza del producto.

COSTES TOTALES		
Elementos		**Coste total parcial**
Robot		47,85
Cámaras		49,03
Soporte Camara		5,62
Otros		5
Mano de obra		4910
Costes indirectos		752,63
Total costes de fabricación		5770,13
Beneficio sobre coste (10%)		577,02
Coste total del proyecto		6347,15

El coste directo de desarrollo completo del proyecto asciende a la suma de seis mil trescientos cuarenta y siete euros con quince céntimos euros (6347,15 €).